uc3m | Universidad Carlos III de Madrid

Máster Universitario en Ingeniería Industrial

Curso académico 2019-2020

Trabajo Fin de Máster

SIMULACION DE TRAYECTORIAS PARA QUADCOPTERS USANDO FAST MARCHING

Francisco Javier Galarza Sánchez

Tutor

Luis Santiago Garrido Bullón

Leganés, Julio 2020

Resumen

El objetivo propuesto para este proyecto es el desarrollo de un sistema capaz de simular vuelos con drones quadcopters. Estas simulaciones se pretenden llevar a cabo en entornos urbanos, cuyas trayectorias se deberán calcular mediante el empleo del algoritmo Fast Marching. Este sistema de simulación debe ser capaz también de poder ser modificado para adaptarse a distintas situaciones de forma que tenga un amplio abanico de posibilidades de simulación.

Otro requisito que deben cumplir las simulaciones es que deben de poder realizarse de forma simultánea vuelos de varios drones. Estos deberán coexistir e interaccionar entre ellos como lo hacen con el propio entorno, definiéndose así las debidas interacciones.

El último aspecto que se considera fundamental para el desarrollo del proyecto es que, además de poder visualizar las simulaciones en tiempo real cuando se ejecutan, de deben poder analizar los datos posteriormente. Toda la información relevante que maneja el sistema debe ser guardada, ordenada y representada para permitir un análisis cuantitativo posterior a la propria simulación.

Durante el desarrollo de esta memoria se pretende explicar los fundamentos que han hecho posible la construcción del sistema descrito. Aunque sin entrar en los detalles a nivel de programación, se pretende dar un entendimiento general del funcionamiento de todos los sistemas que componen el simulador final.

La parte final del documento se enfoca a mostrar los resultados obtenidos con el producto final. En ella se mostrarán y analizarán los valores obtenidos para de esta manera poder comprobar si el funcionamiento se ajusta a los requerimientos establecidos.

Abstract

The proposed objective for this project is to develop a system capable of simulating quadcopter drones' flights. These simulations are planned to be happening in urban environments, where paths must have been calculated using the Fast Marching algorithm. The simulation system must also be able to be modified to adjust the initial conditions of the flight to simulate a wide range of cases.

Another requirement of the simulation is that there should be multiple drones flying at the same time. Those drones must coexist and interact with themselves and their surroundings to archive each desired destination.

The last aspect considered to be essential to the project is not only being able to visualize simulations in real-time but also to be capable of analyzing data after the simulation has ended. Every piece of relevant information must be saved, processed, and organized to allow a deep quantitative subsequent analysis.

Through this document, the basis that has allowed to develop the project will be explained. Although it is not intended to explain every detail of the code, there should be enough content to have a general understanding of each part that contributes to the final simulation system.

The last sections of the document will be focused on showing the results archived in the final version of the simulator during simulations. In those sections, results and his analysis will be displayed to check if the workings are all in the agreement of the previous objectives and requirements.

Índice de contenidos

Índice de figuras

Índice de tablas

CAPÍTULO 1: Introducción

1.1 Motivación

En los últimos años el desarrollo y uso de los drones está sufriendo un gran auge debido a las nuevas posibilidades que aportan. La posibilidad de realizar vuelos en las tres dimensiones del espacio simultáneamente sin la necesidad de ninguna infraestructura previa, junto con la capacidad de mantener una posición y orientación deseada constituyen su mayor ventaja respecto a sistemas alternativos. Los drones representan una oportunidad de transporte rápido y flexible con capacidades de vuelo autónomo lo que permite adaptarse a casi cualquier situación.

Estos sistemas aéreos gracias a tener la capacidad de portar carga, se pueden emplear en tareas de transporte de pequeños objetos o llevar consigo multitud de sensores que les permitan navegar de forma autónoma o recoger datos para su posterior análisis.

Todas estas nuevas operaciones traen consigo la necesidad de realizar pruebas previas a su implementación con modelos físicos, especialmente en los vuelos autónomos. Es en este punto donde la simulación se convierte en el primer y más importante punto en el desarrollo de estos sistemas, debido a que permiten comprobar el comportamiento en entornos variados durante la fase de desarrollo.

1.2 Objetivos

El objetivo de este proyecto es planificar misiones para quadcopters, o llamados drones por simplicidad, para posteriormente poder realizarlas en entornos urbanos simulados. Para ello, la primera de las metas será la definición del entorno de la misión que serán altos edificios presentes en las urbes.

En este entorno se pretende calcular las trayectorias que deben seguir de forma autónoma haciendo uso del método de Fast Marching para obtenerlas. Estos recorridos se pretende que sean lo más seguros posibles en cada caso. Teniendo en cuenta los obstáculos cercanos y las posiciones de otros drones que se encuentren en las proximidades, se calcularán los itinerarios que permitan una navegación más sencilla.

Tras el cálculo de estas trayectorias será necesario la implementación de un sistema capaz de ejecutarlas, este será el sistema controlador. Se encarga de ajustar en tiempo real las posiciones de cada dron para que realice las trayectorias de la forma más fiel posible al plan previo.

Por último, se requiere que la simulación sea fácil de visualizar y comprender el comportamiento de cada dron durante el proceso. Además, se tendrá que registrar los datos obtenidos a fin de poder mostrarlos y analizarlos tras la simulación y determinar el éxito relativo de la misión.

1.3 Estructura

La estructura interna de la memoria se muestra a continuación junto con una pequeña descripción de los aspectos más importantes abordados en cada uno de los apartados.

- **Estado del arte:** Una breve introducción al estado actual y los aspectos más relevantes del mundo de la robótica para el proyecto. Se incluyen los temas de planificación de trayectorias y la robótica en general.

- **Recursos empleados:** Se trata de un listado y pequeña descripción de todos los programas que han sido necesarios para la realización del proyecto. Se encuentran divididos entre dos categorías dependiendo del sistema operativos a los que pertenecen.

- **Diseño e implementación:** Es el cuerpo del proyecto. En este apartado, se describe el proceso que se ha seguido para conseguir el simulador final. Se comentan las opciones contempladas, así como la solución final para cada componente.

  Este capítulo se encuentra estructurado en tres partes bien diferenciadas. La primera se centra en la obtención de las trayectorias, la segunda en conseguir un entorno donde poder desarrollar la simulación, y la tercera en obtener un sistema de control que sea capaz de que los drones recorran las trayectorias propuestas dentro del entorno de simulación creado.

- **Resultados del proyecto:** En este cuarto capítulo se exponen los resultados obtenidos con el simulador. Se tratan varios ejemplos de misiones con distintas características que sirven para ajustar, visualizar y comprobar parámetros de funcionamiento. Adicionalmente se modificarán sistemas previos para ajustarse a las necesidades de cada simulación y demostrar así las capacidades que tiene el simulador completo. Otro objetivo adicional es el de mostrar la capacidad posterior de análisis de los datos recogidos.

- **Marco socioeconómico:** En este apartado se tiene un resumen de la fase de preparación y planificación del proyecto, así como una propuesta de presupuesto para el proyecto.

- **Conclusiones:** Este capítulo recopilatorio recoge los aspectos conseguidos, valorándolos en el conjunto de todo el proyecto.

- **Trabajos futuros:** El contenido de este capítulo se centra en los aspectos que por su extensión, complejidad o por no estar considerados inicialmente no se han realizado, pero que tienen el potencial de convertirse en expansiones apropiadas al proyecto.

- **Bibliografía:** Este último capítulo recoge las fuentes empleadas para la documentación que ha sido necesaria para realizar el proyecto.

CAPÍTULO 2: Estado del arte

En este segundo capítulo se abordan dos ámbitos que resultan fundamentales para la comprensión del proyecto. El primero es la planificación de trayectorias en general, y más en detalle el algoritmo empleado en este proyecto en concreto. La segunda parte se enfoca al origen de la robótica y las necesidades de entornos de simulación que ha traído consigo.

2.1 Planificación de trayectorias. Método de Fast Marching

El cálculo o planificación de trayectorias es un problema muy estudiado no solo en el campo de la robótica, sino que se aplica a aspectos tan diversos como las inteligencias artificiales o los videojuegos. Debido a que no es un problema nuevo, y que las matemáticas desarrolladas para solucionarlas están bien caracterizadas, este problema se considera resuelto. La obtención de una trayectoria que una un punto de inicio y otro de final teniendo en cuenta los obstáculos que puedan estar presentes en el entorno, es una tarea para la que se tienen numerosas maneras de proceder.

Sin embargo, lejos de abandonar esta problemática, se siguen desarrollando y mejorando algoritmos. Esto se debe a que, aunque es fácil obtener trayectorias, el enfoque actual es en conseguir un óptimo que proporcione el recorrido más rápido y seguro. Es aquí donde existen gran multitud de sistemas que, según su forma de operar generalmente se clasifican en dos grupos, los de planificación por combinación, o *Combinatorial planning* y los de planificación por muestreo o *Sampling-based planning*.

Los métodos de *Combinatorial Planning* se basan en descomponer el entorno de posibles soluciones en pequeños segmentos. Estos segmentos se combinan tratando de obtener una solución que permita conectar los dos puntos seleccionados. La salida de este tipo de algoritmos debido a su naturaleza completa, tendrá siempre una salida definida y, por tanto, se podrá conocer con seguridad si existe o no una solución. [1]

El segundo grupo, el de los *Sampling-based planning*, tiene un enfoque de prueba y error en el que se generan distintos caminos o paths aleatorios de los que se seleccionan aquellos que se acercan más a la solución. Este proceso se repite de forma iterativa hasta que el algoritmo alcanza el objetivo. Esta categoría de búsqueda a diferencia de la anterior no se considera completa. Esto quiere decir que, si existe solución, el sistema la encontrará en un tiempo finito, pero si por el contrario no existiera, el sistema no llegaría nunca a esa conclusión pues nunca será capaz de comprobar las infinitas posibilidades que este método implica. [2]

El método empleado en este trabajo es el algoritmo de Fast Marching, inicialmente desarrollado por James Sethian. Se engloba dentro de la segunda categoría descrita, los métodos de planificación por muestreo. Se ha seleccionado por su capacidad de encontrar rutas más seguras y eficientes, aspecto que se considera indispensable para el vuelo de drones.

El concepto intuitivo tras el funcionamiento de este método es frecuentemente comparado con el fenómeno de refracción, donde la velocidad a la que la luz viaja depende del medio por el que lo hace. Esto se implementa en el método de Fast Marching de forma que la velocidad máxima permitida en zonas adyacentes a obstáculos es nula, y aumenta a medida que la distancia a ellos aumenta a medida que se aleja de la frontera. [3] [4]

Empleando esta simple regla, se puede crear un mapa del entorno de la simulación con los distintos índices de refracción o velocidades máximas asociados a cada punto. En la figura 2.1 se puede ver el resultado del mapa de velocidades. En color negro se pueden ver los obstáculos o fronteras del entorno, los muros que componen el espacio de la simulación. Se puede ver como a medida que la distancia a ellos crece el color del mapa presenta una transición de grises hasta llegar a colores más blancos que denotan las zonas de velocidad máxima. La frontera está representada en color negro porque son zonas que no se pueden atravesar, por lo que la velocidad debe ser nula.

Figura 2.1 Mapa de velocidades del método Fast Marching [5]

Mediante este mapa se puede visualizar la variación que experimenta el factor que limita las velocidades. Generalmente su valor varía de desde 0 hasta 1 o hasta 255, lo que se usa para asignar a estas variaciones los colores correspondientes con la transición blanco-negro en la representación en escala de grises.

El método de Fast marching una vez tiene el mapa de velocidades, se encarga de generar una onda desde el punto de inicio de la trayectoria desde donde se expande en todas direcciones, siempre respetando las velocidades máximas determinadas por el mapa anterior. De esta forma, el frente de onda avanza a distintas velocidades dependido de la zona en la que se encuentre, quedándose estancado en zonas adyacentes a obstáculos y avanzando a mayor velocidad a medida que la distancia a ellos aumenta.

Con el avance de los frentes de onda, llega un punto en el que se termina por alcanzar el punto de destino deseado. Es en este momento cuando el algoritmo ha conseguido establecer la solución al problema planteado. La trayectoria final es el resultado de unir los frentes de onda que han permitido alcanzar el destino deseado. La representación gráfica de esta solución se puede ver en la figura 2.2.

En ella se puede apreciar cómo, aunque solo existe un frente de onda que ha alcanzado el destino, debido a la geometría del entorno se han creado varios frentes que han divergido del frente solución. En la parte superior izquierda y en la inferior izquierda se pueden apreciar dos frentes que no han terminado de expandirse cuando el algoritmo ha finalizado. Sin embargo, estos no son los únicos, se puede ver como en las salas que se encuentran cercanas al punto inicial, la onda se ha expandido completamente, aunque no ha llegado al destino requerido. Es aquí donde se pone de manifiesto este carácter iterativo del algoritmo, donde se van descartando y ampliando ramas a medida que la simulación avanza.

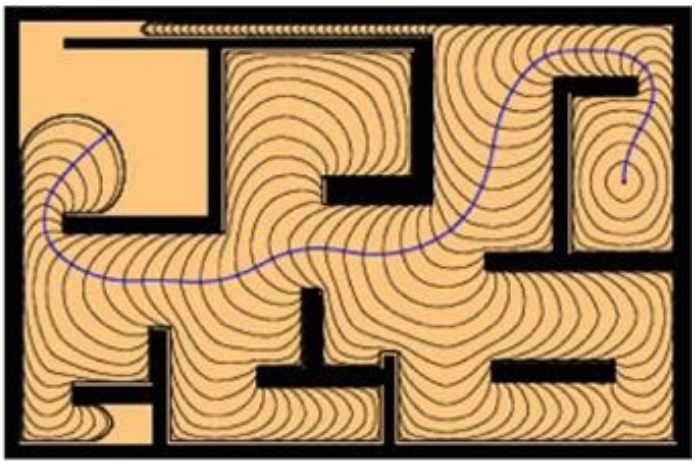

Figura 2.2 Frentes de onda y trayectoria final como resultado de ejecutar Fast Marching [5]

Este método tiene distintas variantes entre ellas la conocida como FM^2 o *Fast Marching Square*, que modifica el mapa de velocidades obtenido. Cuando las necesidades de seguridad del trayecto no requieren de una distancia muy grande los obstáculos, se puede saturar el mapa. Esto significa delimitar una distancia determinada a partir de la cual la velocidad permitida sea la velocidad máxima seleccionada.

2.2 Desarrollo de la robótica. Herramientas de control y simulación

Los sistemas autómatas son un tema que lleva fascinando a la humanidad desde sus origines. Desde los primeros y simples sistemas de válvulas y poleas que dotaban de movimiento a objetos con el mero fin de crear la ilusión del movimiento de objetos inanimados.

El primer ejemplo de la era moderna sería el escritor, una invención de Jaquet-Droz una pequeña figura con apariencia de niño que mediante sus mecanismos internos aparentaba estar escribiendo una carta. Sin embargo, no es a partir de la segunda mitad del pasado siglo cuando la robótica ha tomado un acercamiento más práctico, con el desarrollo de los primeros brazos robóticos, estos ya empleados para fines más prácticos como es la asistencia en las líneas de producción o manipulación de objetos peligrosos.

En los últimos años este crecimiento y evolución ha sido algo exponencial, y ha afectado no solo al ámbito de la producción existiendo líneas totalmente automatizadas con escasa intervención humana, sino también en el ámbito de la investigación e incluso llega a ser objeto de entretenimiento para algunos aficionados.

Esta mayor capacidad de los sistemas robóticos ha llevado consigo la necesidad de sistemas de control cada vez más complejos, que sean capaces de, no solo controlar, si no también deben decidir en función su entorno las respuestas más adecuadas en tiempo real. Es de esta necesidad donde surgen los sistemas como ROS.

ROS, *Robot Operating System*, o Sistema operativo para robots, es la solución de software abierto que lidera los procesos de investigación y desarrollo en el campo de la robótica abierta. Se trata de un entorno con gran variedad de herramientas y librerías destinadas a todo lo relacionado con su control y sus flujos de información. El hecho de que sea software abierto hace

que este en constante evolución y mejora, ya que cualquiera puede contribuir a mejorarlo y esto es algo que ocurre constantemente con las aportaciones de investigadores de todo el mundo.

La posibilidad de crear robots cada vez más complejos hace necesaria la aparición de entornos que permitan probar estos sistemas antes de su implementación física. Los simuladores permiten la recreación de no solo el sistema a ensayar, sino también las condiciones del entorno en el que se va a emplear. De esta forma se permite mejorar el diseño de los robots y encontrar fallos en el funcionamiento antes del proceso critico de fabricación de prototipos.

No obstante, esta no es la única forma de emplearlos, también se pueden utilizar para simular misiones completas en las que comprobar si el comportamiento mostrado es el deseado en situaciones que pueden variar desde uso normal hasta algunas extremas para así poder realizar pruebas sin poner en riesgo el equipo físico.

Aún con todas las ventajas de la simulación, esta no puede llegar a reemplazar las pruebas físicas, solo supone un primer y necesario paso en el desarrollo que puede agilizar futuras pruebas en los entornos finales de trabajo donde se debe comprobar su funcionamiento final.

CAPÍTULO 3: Recursos empleados

Para la realización de este proyecto ha sido necesario usar una amplia selección de programas y aplicaciones que se van a listar y brevemente describir en este capítulo para poder contextualizar mejor el trabajo.

Debido a la variedad que existe en los programas empleados, se ha requerido usar simultáneamente dos sistemas operativos distintos, Windows y Linux.

3.1 Entorno Windows

En el sistema operativo Windows se ha hecho uso de Matlab como la principal herramienta de programación, además de las aplicaciones de virtualización y modelado 3D.

- **Matlab:** es un software orientado tanto a la investigación y académicos como a aplicaciones profesionales. Cuenta con su propio lenguaje de programación, y su principal ventaja en este proyecto es la gran variedad de librerías disponibles en el ámbito de la robótica. Con estas se puede conseguir de forma relativamente sencilla la integración de los distintos programas necesarios. Además, cuenta con ejemplos de programación en la mayoría de estos recursos, lo que permiten una fase inicial del proyecto más rápida.

  Merece especial mención el entorno de **Simulink** como aplicación disponible dentro de Matlab. En ésta se puede realizar la programación en un entorno que, aunque contiene las funcionalidades clásicas de programación, también permite la programación mediante interfaces de bloques de función.

- **VMware Workstation:** se trata de un software que trabaja como máquina virtual dentro del entorno Windows. Es capaz de emular un sistema independiente operado por distintos sistemas operativos. De esta forma se permite tener acceso desde un mismo equipo de forma simultánea a distintos sistemas operativos con la versatilidad que ello conlleva.

- **Blender:** este es un programa gratuito de software abierto enfocado al modelado 3D. Será necesario para la creación y modificación de modelos tridimensionales que se representen el entorno dentro del simulador.

3.2 Entorno Linux

Para el sistema operativo de software abierto Linux se ha hecho uso de la distribución Ubuntu 14.04.4, instalada en el mismo equipo mediante la máquina virtual mencionada. En este entorno los programas empleados se listan a continuación.

- **ROS:** es la aplicación que permite el control de los quadcopters empleados. Es también el encargado de definir y gestionar los flujos de información desde y hacia cada dron, por lo que toda acción que se requiera debe usar a este programa como nexo de comunicación.

- **Gazebo:** se trata del entorno de simulación principal del proyecto. Este programa se encarga tanto de mostrar el entorno modelado para la misión, como los de los propios drones. También se encarga de modelar el comportamiento mediante el simulador de físicas, que tiene en cuenta no solo las fuerzas externas que se aplican, sino también las provenientes de los propios drones. Gracias a esto se puede tener una representación realista de las interacciones entre elementos.

- **Kate:** es un programa de edición de texto avanzado. Este permite modificar los archivos necesarios para hacer funcionar la simulación. Gracias a sus herramientas de tabulación y reemplazamiento de texto hace que sea mucho más sencillo trabajar y ordenar los documentos con código necesarios para la ejecución de las simulaciones.

CAPÍTULO 4: Diseño e implementación

La implementación de este proyecto tiene la complicación adicional de que hace uso de gran cantidad de programas distribuidos en distintos sistemas operativos. La organización interna que tiene el proyecto junto con sus conexiones se puede ver representada en la figura 4.1. En ella se puede apreciar como todo queda englobado bajo el sistema Windows con Matlab y Simulink que se comunican con el sistema en Ubuntu mediante ROS, que se conecta con el simulador Gazebo. Todos estos flujos de datos se realizan en ambos sentidos para asegurar que se lee y envía la información en los sistemas correctamente.

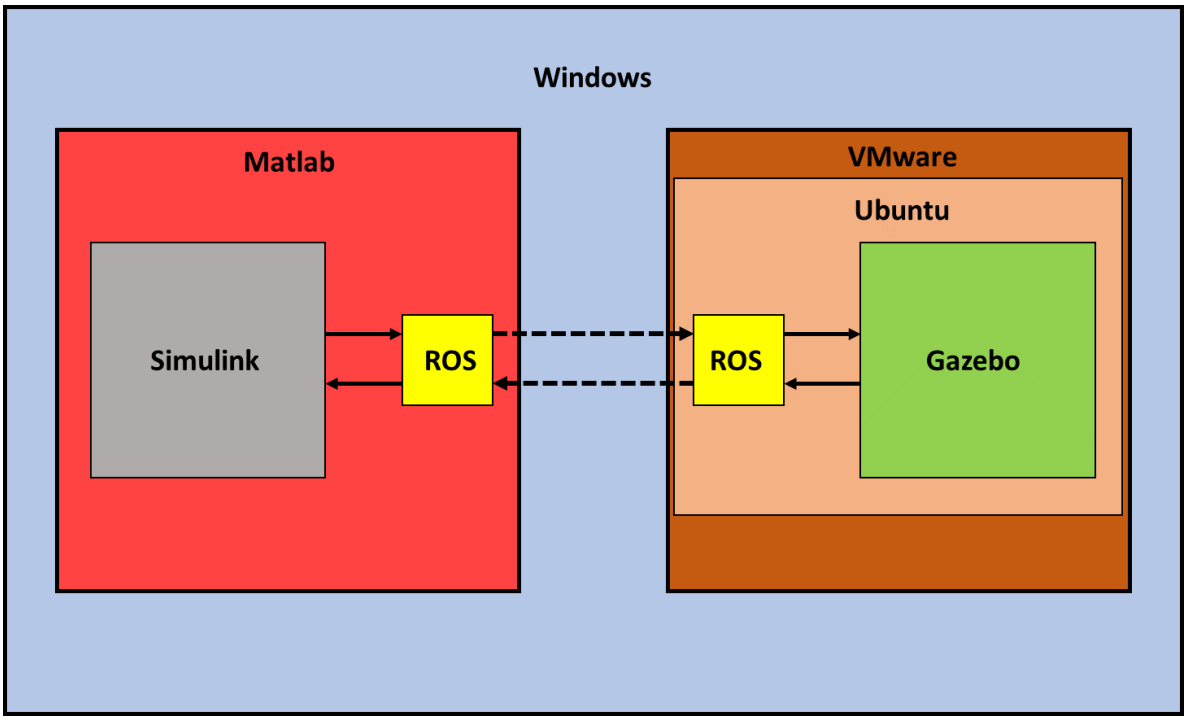

Figura 4.1 Esquema de las conexiones entre aplicaciones

En los siguientes apartados se va a desarrollar todas las bases del diseño y su implementación atendiendo al propósito que sirven dentro de la aplicación final. Según esto, las tres secciones son: la programación y previsualización de las trayectorias, la configuración del entorno final de simulación, y como ultima sección, se tiene el diseño del sistema de control de los distintos drones.

4.1 Obtención de las trayectorias

En esta sección se va a describir la parte centrada en el uso del algoritmo de Fast Marching para el cálculo de las trayectorias para los distintos drones. Además, se realiza una simple previsualización de este resultado antes de pasar a la simulación final para poder confirmar de una forma sencilla que los parámetros y cálculos se han realizado correctamente.

4.1.1 Implementación del algoritmo

Como se ha comentado anteriormente, la implementación del algoritmo de Fast Marching se ha realizado en Windows usando Matlab. Para ello se ha hecho uso de una implementación del algoritmo creado para la simulación de vuelos con múltiples drones de forma simultánea. En esta implementación además de calcular las trayectorias más favorables mediante Fast Marching, se permite la definición de distintos parámetros como son el número de drones, los límites del entorno, los edificios que conforman los obstáculos o la prioridad de los distintos drones. [6]

La implementación permite definir las posiciones de inicio y final mediante vectores con las tres componentes cartesianas que las definen, mientras que el entorno de la simulación se define mediante una matriz tridimensional binaria, donde solo se diferencian las zonas ocupadas con obstáculos del resto del espacio libre del que pueden hacer uso los drones para desplazarse.

El cálculo de las trayectorias mediante este algoritmo en cada dron trata al resto de ellos como obstáculos y, por tanto, trata de evitar trayectorias que pasen cerca de ellos. De esta forma el algoritmo tiene preferencia por aquellos itinerarios que, aunque más largos, permiten un menor tiempo de llegada al objetivo final gracias a su mayor velocidad permitida en estos recorridos alternativas, que las busca haciendo uso de su naturaleza iterativa.

El resultado que se obtiene tras introducir todos estos parámetros y completar el cálculo son los vectores de coordenadas que contienen todos los puntos de las trayectorias por las que cada dron debe desplazarse de forma ideal, su trayectoria óptima.

No obstante, estos vectores solución no nos permiten con solo mirarlos determinar si los parámetros obtenidos y, por ende, la solución es la deseada. Por este motivo es necesario realizar distintas visualizaciones, entre la que la más importante es la tridimensional, a fin de comprobar su adecuación. Esto es importante realizarlo antes de pasar a realizar la simulación final, ya que ésta requiere de mayor tiempo de ejecución y una puesta en marcha más compleja que volver a ejecutar este algoritmo con los nuevos datos deseados.

4.1.2 Implementación de la previsualización de trayectorias

Las características principales de esta previsualización deben ser su claridad y detalle de representación y la rapidez en su obtención.

Una claridad y un detalle suficientes que permitan diferenciar los distintos elementos, obstáculos del entorno, drones y trayectorias, comprobando si se adecúan a las especificaciones previas establecidas. Además, el tiempo de ejecución debe ser más reducido, mucho menor al tiempo de ejecución real que se requerirá en la simulación final, para así facilitar rápidas iteraciones en las condiciones simuladas.

Para realizar esta representación, se puede comenzar por la parte más voluminosa, el entorno urbano, que se compone principalmente por los edificios. Para representarlos, se sopesaron distintas alternativas, de las que se van a comentar las que mejores resultados han ofrecido.

La primera de ellas es la que hace uso de las *Isosurfaces*, esto es, una representación del entorno mediante una única superficie continua que envuelve todos los obstáculos presentes en la representación. El resultado del entorno obtenido haciendo uso de esta técnica se puede ver en la figura 4.2.

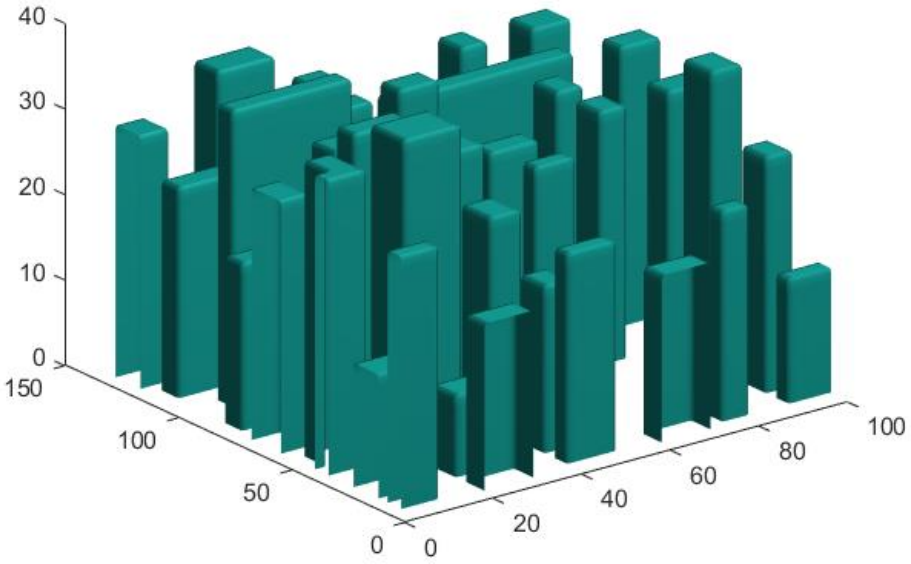

Figura 4.2 Representación del entorno mediante isosurfaces

Esta representación permite tener una primera idea del aspecto del entorno, donde se pueden distinguir los distintos prismas que simulan el escenario urbano. Para la mayor parte de las pruebas realizadas durante el proyecto, el entorno urbano ha sido el mismo, ya que cambiarlo con las iteraciones no aporta ninguna ventaja en esta etapa de desarrollo, mientras que mantenerlo constante proporciona un marco de referencia para las iteraciones en el diseño.

El entorno se compone por un total de 41 edificios, con distintas alturas, separaciones y tamaños dependiendo repartidos por toda la superficie de 100 por 150 metros. De esta manera, se pueden observar distintos comportamientos dependiendo de las zonas empleadas en la simulación.

A pesar de ser una buena primera aproximación, se aprecian ciertas limitaciones en esta implementación. Aunque el método de representación cuenta con multitud de parámetros de ajuste y es lo suficientemente rápido, carece de la capacidad de ajustar estos parámetros de forma local, ya que se aplican de forma uniforme a toda la representación. Por este motivo, no se puede conseguir una claridad suficiente para poder distinguir con facilidad unos edificios de otros.

El principal parámetro que afecta en gran medida es la transparencia, y con este método no se puede asignar un valor que tenga el equilibrio adecuado entre visibilidad de las trayectorias tras los edificios y poder distinguir estos con claridad. Aplicando este sistema, la configuración que se ha mostrado, con la opacidad al máximo, es la que más nos acerca a un resultado adecuado. Es así, que esta alternativa se terminó descartando en favor de otra que permitiera un mayor control de los parámetros de forma local.

La alternativa que finalmente se terminó empleando es la función *Fill3*. Este comando permite la creación de polígonos coplanarios definidos por sus vértices. Este método cuenta con el mismo nivel de parametrización que el método, pero con la diferencia de que se pueden asignar valores de forma individual a cada uno de estos polígonos.

Para emplear este sistema es necesario mediante programación extraer de la matriz donde se define el escenario los 8 vértices que componen cada uno de los edificios. Esto se ha hecho mediante programación en lugar de usando las definiciones del entorno para asegurar que posteriores cambios en el entorno permitan seguir representándolo de forma automática. Estos 8 vértices se agrupan de forma que se formen 5 caras para cada prisma, las cuatro caras laterales y la cara superior. La base inferior no se ha representado para no crear polígonos innecesarios, y se ha reemplazado por un único rectángulo que abarca toda la altura cero del espacio representado.

Para la configuración de estos edificios se ha decidido representar las caras paralelas a cada eje (el X e Y) con el color típicamente asociado, rojo para el eje X, y verde para el eje Y; para el caso del eje Z las caras que son perpendiculares en este caso se han representado en azul. En las caras opuestas de los ejes X e Y se han usado sus colores correspondientes, pero ligeramente más claros con el propósito de poder diferenciar de forma visual la zona observada de la representación sin tener que recurrir a mirar los ejes presentes en los márgenes del modelo tridimensional.

El otro parámetro importante además del color es la transparencia de las caras. Esta se ha establecido para las caras laterales frontales en un valor de bastante opaco. Con un factor de 0.8 sobre 1 en opacidad y de 0.7 para la cara superior que permite algo más de transparencia. Este valor permite intuir otros edificios tras ellos y posteriormente tras añadir las trayectorias ha permitido visualizarlas a pesar de estar parcialmente ocultas tras ellos.

Para las caras posteriores se emplea un valor de opacidad bastante menor, de 0.2 lo que permite tener una visión más transparente si se observa desde la parte trasera. Además, esto facilita en gran medida la visibilidad desde la parte frontal de la representación ya que son menos visibles y hace más fácil poder distinguir los objetos que se representen tras ellos, aunque lo suficiente para que puedan distinguirse si se observan directamente.

El resultado final del entorno aplicando estas modificaciones se puede ver en la figura 4.3. Además de las mejoras visuales mencionadas, se puede notar una definición más precisa de las aristas de los prismas, que en este caso no son redondeadas como ocurría con la aplicación del método anterior.

Esta representación, no es una mera imagen fija, sino un modelo tridimensional completo, por lo que en el entorno de Matlab se puede rotar y ampliar para ver zonas con más detalle, o para visualizar aquellas zonas en las que por la cantidad de edificios sería difícil observar en un plano más general.

Una vez se tiene definido este entono tridimensional, es necesario representar también las trayectorias de los distintos drones. Las pruebas se han realizado casi por completo con 3 drones con el propósito de tener un número reducido con el que comprobar el funcionamiento general, pero suficiente para poder comprobar iteraciones y simultaneidad. Sin embargo, este número puede ser aumentado fácilmente añadiendo los parámetros necesarios en la definición del problema.

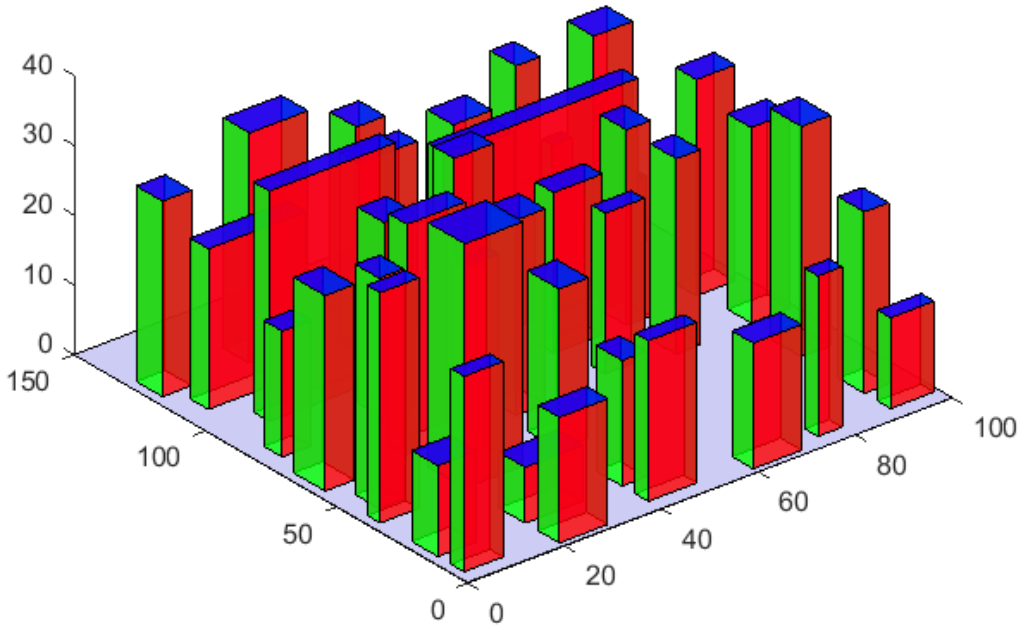

Figura 4.3 Representación del entorno mediante Fill3

Para representar sus trayectorias, se ha hecho uso de la librería *Flypath3D*. Se trata de un conjunto de utilidades destinadas a la representación de vehículos aéreos y sus trayectorias.

Para ello como en cualquier implementación de un nuevo sistema, se deben adaptar los datos de las trayectorias para que sean legibles por las funciones de la librería. Además, se debe importar el modelo del dron que se pretende emplear, en este caso al ser un quadcopter se trata de un dron de cuatro hélices. Aunque esta librería cuenta con la opción de importar modelos propios, por simplicidad se ha hecho uso de uno de los modelos que encajan con la descripción de los quadcopters empleados.

La librería cuenta con dos opciones para la representación, una secuencial donde el recorrido que realizan los drones se representa mediante una animación, y otra que permite mantener de forma fija la trayectoria del trazado recorrido por cada dron con representaciones del vehículo en puntos intermedios. Para la finalidad de análisis, la segunda opción es mucho más adecuada, al igual que para el propósito de la redacción de esta memoria, por lo que esta será la opción seleccionada. [7] [8]

En las dos imágenes que se encuentran a continuación, se puede ver el resultado de la previsualización de las trayectorias con 3 drones en el entorno. En la figura 4.4 se puede ver una vista frontal en la que se aprecian las trayectorias de los tres drones con varias de las posiciones intermedias del recorrido por las que pasan.

Los drones no son modelos a escala, ya que serían demasiado pequeños como para poder observarlos, sino que son representaciones que únicamente pretenden simbolizar el tipo de vehículo empleado en la simulación.

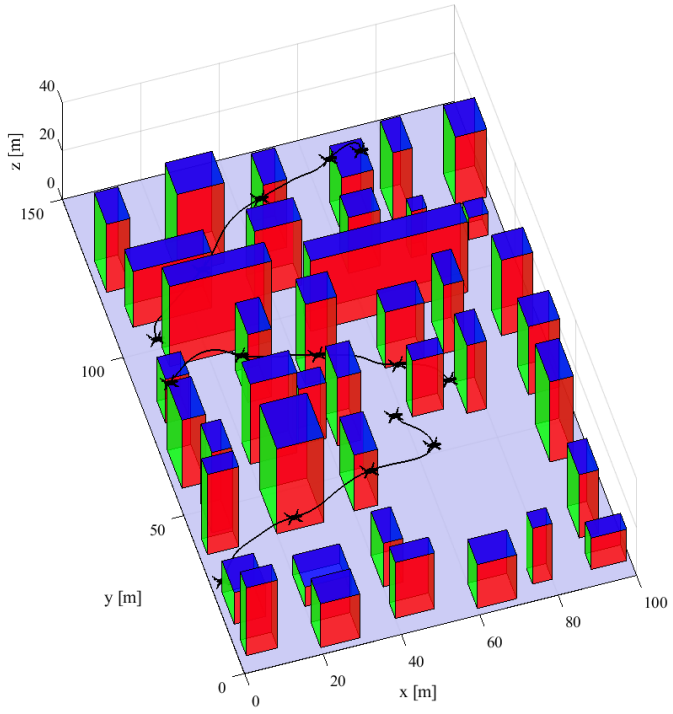

Figura 4.4 Previsualización completa desde el origen de coordenadas

La figura 4.5 por su parte muestra la misma representación desde otro punto de vista. Tiene las caras principales del eje X visibles, pero muestra la parte posterior de las del eje Y por lo que el color verde que tienen los edificios es menos intenso. En esta figura se puede apreciar la diferencia de color entre ambas caras paralelas al eje, y como estas tienen una mayor transparencia permitiendo una visualización más clara de los elementos ocultos, a la vez que permite identificar que en la perspectiva se observa la parte posterior del eje Y.

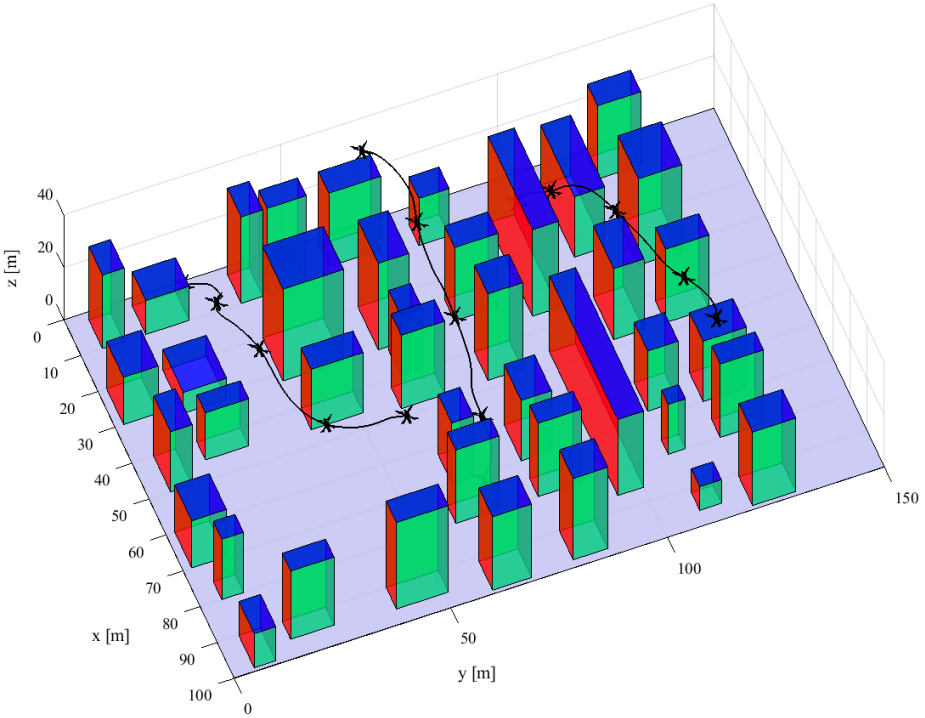

Figura 4.5 Previsualización completa desde la parte posterior del eje Y

4.2 Entorno de simulación

En esta sección se va a tratar el desarrollo del entorno de simulación final, que como ya se ha comentado se ha realizado de forma exclusiva en el sistema Ubuntu Linux en su versión 14.04.4, que aunque no es la más actual, es una de las versiones que se recomiendan desde algunos proyectos similares en Matlab. Para ello, se ha instalado en la máquina virtual el sistema operativo junto con ROS, el entorno de simulación, GazeboSim en su versión 2.2.3, además de un editor de texto avanzado como es Kate. [9]

La sección está subdividida por funcionalidades, de esta forma se explicará en orden cronológico las funciones que se han implementado para tener el entorno de simulación operativo.

4.2.1 Configuración general de Gazebo

Gazebo es un simulador enfocado al trabajo con robots móviles y articulados. En este entorno es sencillo crear este tipo de robots incluso mediante formas geométricas como cubos, esferas y cilindros y relacionarlos mediante uniones articuladas que permiten giros relativos entre los elementos. Adicionalmente en la descripción de los sistemas se pueden modificar parámetros como el rozamiento, la masa, la colisión o la visibilidad de los distintos componentes para poder facilitar la creación de una gran variedad de sistemas.

El robot más simple que se suele emplear para multitud de ejemplos en la documentación es el TurtleBot. Éste representa el concepto más básico de robot móvil, tratándose de una plataforma móvil con tracción diferencial. Este tipo de propulsión le permite controlar su movimiento únicamente variando la velocidad relativa de sus dos ruedas motrices, lo que supone únicamente dos parámetros. Generalmente, además de estas dos ruedas suele tener una adicional cuyo único propósito es dotar al sistema de estabilidad y permitir un movimiento estable y fluido, aunque pueden existir distintas configuraciones dependiendo de las modificaciones creadas al TurtleBot. Es habitual que estos robots incorporen todo tipo de sensores y es por este motivo que se suele emplear como plataforma para realizar pruebas con ellos. [10]

Dentro del mundo de la simulación no es diferente, ya que permite realizar las primeras pruebas y asegurarse de que los sistemas de control y recepción de datos funcionan correctamente. Para esto se puede optar por dos alternativas, la de importar un robot completo con sus geometrías y sensores o crear uno desde cero. La segunda opción es viable incluso si su único propósito es comprobar los sistemas ya que se trata de un robot muy simple. Uniendo unos dos cilindros que hacen de ruedas al cuerpo prismático y añadiendo una esfera para dotarlo de estabilidad se tiene su estructura más básica completa. Tan solo es necesario definir las relaciones que existen entre ellas permitiendo el giro en sus articulaciones y se tiene una base móvil completamente funcional. Además, la creación del robot es un buen ejercicio para comenzar a introducirse a la estructura que emplea este simulador. [11]

Ambas versiones del TurtleBot se pueden ver en la figura 4.6. En ella se muestra en la parte izquierda el modelo predefinido con un modelo realista del robot disponible en librerías, mientras que a la derecha se puede ver el modelo creado desde cero con elementos disponibles en Gazebo.

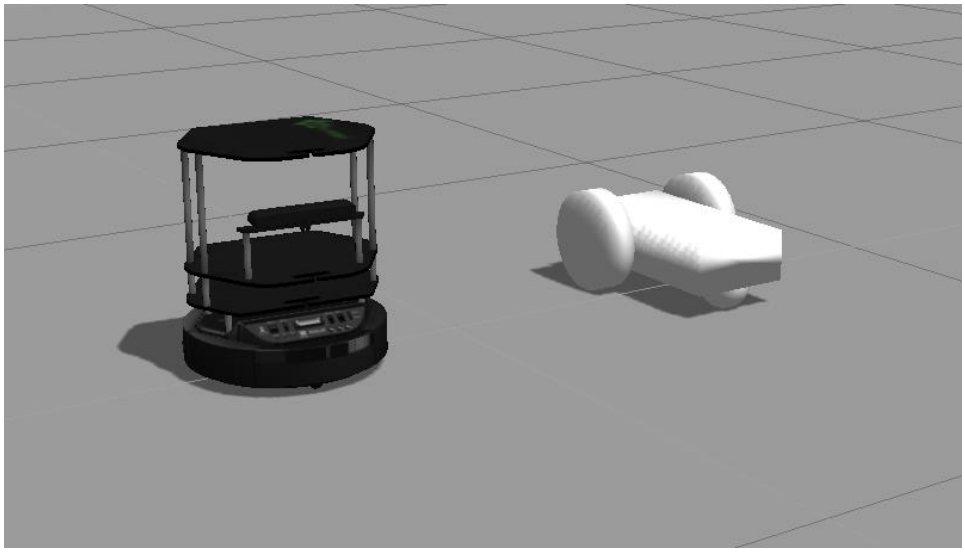

Figura 4.6 Dos versiones del robot TurtleBot en Gazebo

En este caso, aunque la apariencia no es la misma, sus comportamientos en el simulador a efectos de movimiento son idénticos. Esto se puede comprobar ejecutando comandos de movimiento desde Simulink. El proceso de control de los sistemas mediante Simulink se detalla en la siguiente sección: 4.3 Sistema de control. De momento es suficiente con comprobar que con solo las dos entradas que tiene el sistema, las velocidades de cada rueda, se puede controlar el robot y conseguir movimientos en el plano del suelo. El aspecto más importante es que con esta prueba se ha podido configurar y comprobar que tanto la emisión y recepción de datos desde el controlador se realiza de forma adecuado, lo que posibilita seguir avanzando en el proyecto hacia sistemas más complejos.

4.2.2 Implementación de los drones

Una vez se tiene el entorno de simulación y se ha comprobado que funciona correctamente junto con las comunicaciones, se deben reemplazar el TurtleBot por los drones que se pretenden emplear. Para esto gazebo no tiene implementación directa, pues los drones requieren no solo de partes con movimiento relativo como son las hélices, sino también necesitan de la aplicación de fuerzas que simulen el empuje que estas producen. Es por este motivo se requiere de alguna modificación adicional que permita su implementación.

Existen varias alternativas para conseguirlo como son *ardrone_gazebo* o *rotors_simulator*, ambas modificaciones pensadas para implementarlas en ROS y trabajar con Gazebo. Finalmente se optó por hacer uso de hector_*quadrotor*, que cuenta con una funcionalidad muy similar a las alternativas anteriores, pero tiene la ventaja de estar soportado oficialmente por ROS, por lo que se minimizan los posibles problemas de compatibilidad entre versiones.[12] [13] [14]

Esta librería compatible con ROS no solo permite añadir drones funcionales a gazebo, sino que incluye todo un repertorio de herramientas para crearlos y dotarlos de distintas funcionalidades y sensores. Debido a que su función final es la simulación de trayectorias, es suficiente con emplear un dron quadcopter, por lo que, debido a su complejidad, no es necesario crear uno desde cero. Empleando el modelo de dron básico que incluye el paquete hector_quadrotor, definido en el archivo "*quadrotor.urdf.xacro*", que no cuenta con sensores especializados de ningún tipo, se procede a su implementación en el proyecto.

Otra ventaja de usar esta librería es que estos drones de referencia ya tienen implementado un controlador que transforma las ordenes de movimiento en el espacio a los comandos necesarios a cada motor. Este controlador traduce a empuje y a velocidad de giro de cada motor las ordenes de velocidad en los distintos ejes que recibe, simplificando enormemente su control, lo que sin duda es muy útil cuando se diseña el controlador para los drones.

Además, gracias a su integración con ROS también cuenta con una lista de topics básicos. Los topics son nodos que permiten la emisión y recepción de datos entre el controlador y el robot. Estos topics integrados son los más básicos que afectan al movimiento, e incluyen la capacidad de transmitir la posición actual y la de recibir órdenes de velocidad en sus distintos ejes de referencia. Esto se explicará más en detalle en el apartado 4.3.2 Desarrollo del controlador, pero esta es la idea general del sistema de comunicación robot-controlador empleado.

El modelo final del dron empleado visto en el propio simulador se puede ver en la figura 4.7. Se muestra apoyado sobre el plano del suelo en una cuadricula de 1 metro de lado. Las dimensiones del quadcopter son aproximadamente de 0.75 metros de lado en ambas direcciones, y 0.25 metros de altura, con lo que el objeto de mayor interés de la simulación queda totalmente definido.

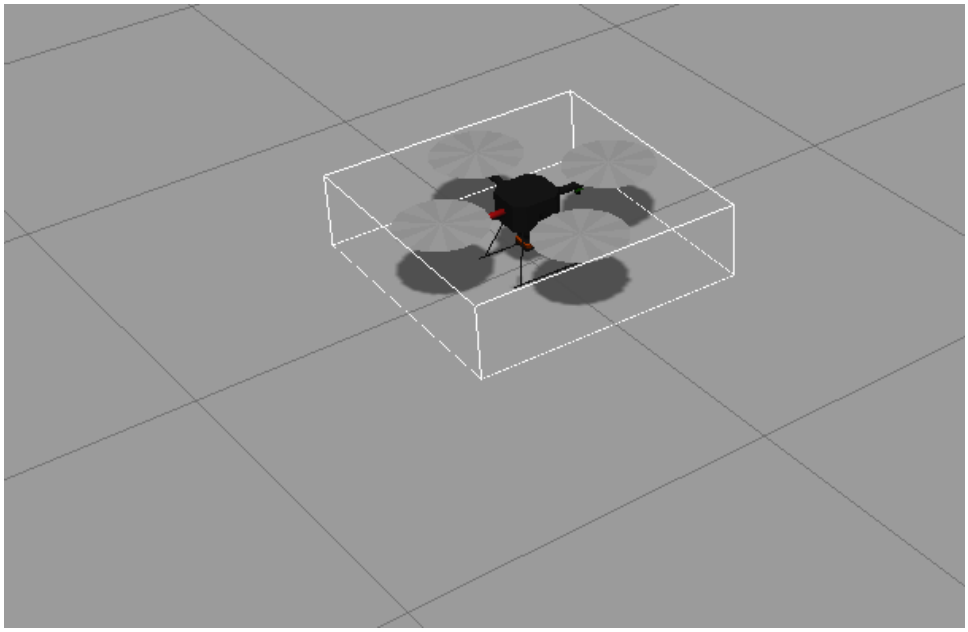

Figura 4.7 Visualización del Dron dentro del entorno de simulación Gazebo

4.2.3 Modelado del entorno

El siguiente paso tras obtener el dron funcional, es la configuración del entorno en el que se va a desarrollar la simulación, en el caso de este proyecto, se trata de un entorno urbano de altos edificios. Para representar este escenario, se ha iterado hasta obtener un entono esquemático lo suficientemente claro para poder identificar los distintos elementos como ha ocurrido en la previsualización en Matlab.

El primer paso ha sido la representación de los edificios, para ello, la primera idea sería hacer uso de los bloques prismáticos más básicos que permite crear Gazebo. Estos prismas se pueden colocar en cualquier parte del entorno, definir sus tres dimensiones de forma manual desde el propio editor del programa. Esta personalización en principio permite que se pueda adaptar el entorno creado al definido en la programación.

Gazebo además permite un grado de parametrización elevado, permitiendo la modificación de distintos parámetros para cada elemento presente. De entre todos los ajustes disponibles, los más reseñables para los obstáculos del proyecto son la condición de elementos estáticos y los elementos visuales. Se selecciona la opción de elementos estáticos por dos motivos fundamentales. El primero es el rendimiento, al no interactuar con las físicas el rendimiento se ve mejorado, además que se evitan problemas con la inicialización de los bloques donde en el caso de interactuar con el plano del suelo en ocasiones y debido a su gran número algunos se tambalean ligeramente.

Además, aunque es posible modificar las dimensiones y coordenadas de los bloques dentro del programa, en la práctica esto no es viable, ya que los ajustes se realizan de forma manual arrastrando los bloques hasta su posición sin la posibilidad de ajustarla a un valor introducido mediante números. Es por este motivo, que para alcanzar una adecuación de los valores se debe hacer modificando directamente los ficheros donde se guardan los detalles y ajustes del mundo. De esta forma se puede tener un mayor control y precisión, ajustando coordenadas y tamaños con sus valores exactos.

Es por esto que se hace necesario el empleo algún editor de texto, en este caso Kate, para poder realizarlo de forma más eficaz. Aunque posteriormente esta edición manual será reemplazada en su práctica totalidad por generación automática mediante programación, al inicio es necesario realizar ajustes manualmente. Debido a la escasa documentación disponible en algunos aspectos, el método de ensayo y error se convierte en ocasiones en la única manera de conseguir los resultados deseados. Es por este motivo que el proceso de automatización queda relegado hasta el momento en el que se conoce exactamente las propiedades que se quieren dotar a cada elemento y con qué sentencias se puede conseguir para implementar la creación del escenario completo.

El siguiente elemento indispensable en la simulación es el plano del suelo. Por defecto gazebo lo crea en todos los mundos, pero debido a las grandes dimensiones del entorno exterior que se pretende representar es necesario modificar sus dimensiones para adecuarlo a las dimensiones del definidas durante el planteamiento del problema.

El último elemento clave en cualquier entorno de simulación es la iluminación. Para obtener una correcta visualización es necesario añadir una fuente de luz adecuada. Debido a que se trata de un entorno exterior se ha empleado una luz direccional, que permite una mayor uniformidad. Este tipo de iluminación se contrapone a la luz puntual y actúa de forma análoga a como lo haría el Sol en la realidad. Además, se ha modificado el color de ésta para mejorar la claridad de la imagen haciendo que emita luz blanca en lugar de una más amarillenta como sería la solar.

El ultimo parámetro de iluminación reseñable es la orientación de esta fuente de luz: en este caso se ha configurado de tal forma que la sombra de los edificios forme unos 45 grados positivos con el eje X, y una posición cercana a lo que podría ser el cenit del día. Esto favorece una iluminación uniforme desde el punto de vista del origen de coordenadas empleado a menudo como referencia de visualización. Además, esta inclinación al generar sombras, pero no demasiado alargadas, hace más sencillo diferenciar entre los distintos edificios, resultando en una mayor claridad de la representación.

El resultado de aplicar todas estas configuraciones del entorno se muestra en la figura 4.8 a continuación.

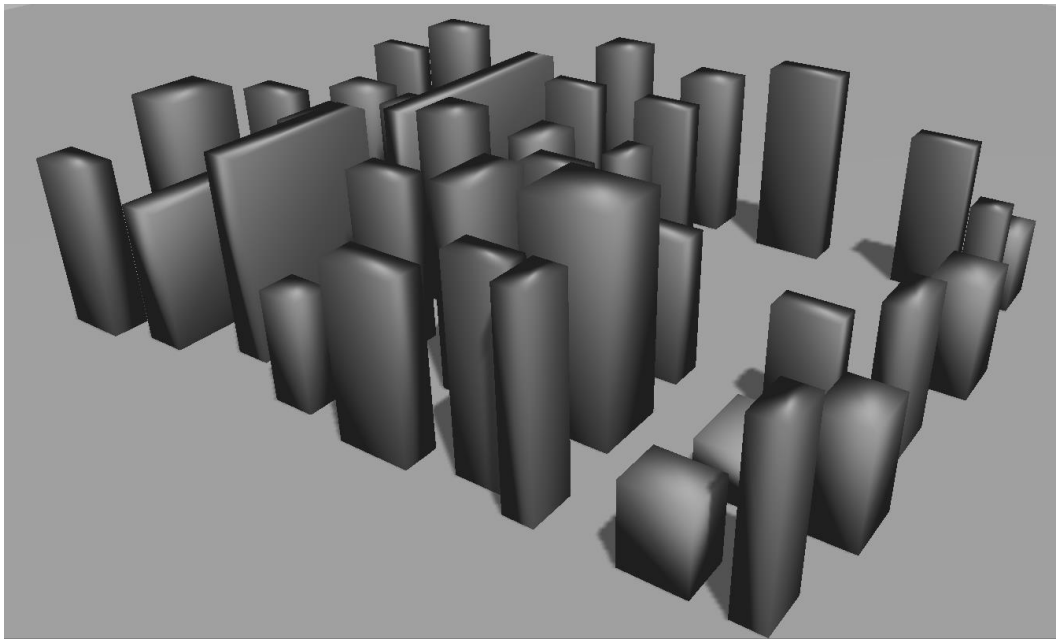

Figura 4.8 Representación inicial del entorno en Gazebo

A pesar de que en esta representación se pueden apreciar los distintos elementos, resulta difícil diferenciar uno de otro en las zonas con mayor densidad, al igual que ocurría inicialmente en la previsualización. Por este motivo y de forma análoga a la representación en Matlab, se pretende cambiar las caras de los prismas para que sean de distintos colores.

Debido a que Gazebo no permite cambiar el color de las distintas caras, sino que solo permite hacerlo de forma general a cada bloque, la forma más eficiente de hacer el cambio es sustituir estos bloques por modelos tridimensionales que tenga los colores correctamente asociados a cada cara. Estos modelos seguirán siendo prismas, para minimizar la carga de cómputo con polígonos innecesarios, aunque podrían modelarse figuras más complejas si esto no resultara un inconveniente y se pretendiera un entorno más realista.

Para realizar estos modelos, se ha hecho uso de Blender, un programa Open Source de modelado 3D disponible tanto en Windows como en Linux. El principal motivo es que es gratuito, lo que lo hace fácilmente accesible, y además permite crear y exportar los modelos en el formato COLLADA (*COLLAborative Design Activity*), formato necesario para que Gazebo pueda interpretar los archivos de forma adecuada.

Para realizar este modelado, únicamente se requiere extruir un cubo genérico de lado 1 metro. Posteriormente durante el proceso de importación a Gazebo se permite el escalado de figuras simples como esta, por lo que se puede adecuar cada una de sus tres dimensiones características. De este modo, se puede obtener un sistema similar al anterior en el que se modifica el escalado del cubo original en lugar de directamente sus dimensiones. Con este modo de construir el entorno se puede conservar la simplicidad de emplear un único modelo para todos los edificios manteniendo la capacidad de adecuarse a las variaciones de tamaño.

Una vez se ha definido el modelo en Blender, es necesario crear los tres materiales necesarios, uno para cada color del RGB, que se corresponderán como en la previsualización con las caras paralelas a los ejes. Estos materiales son colores sólidos que se aplican a las caras correspondientes. Para aplicarlos es necesario crear el mapa de texturas del solido que asocia la ubicación de cada una de estas texturas a las caras correspondientes. El mapa de texturas se puede ver en la figura 4.9, y el resultado final en Blender tras aplicarlo se muestra en la figura 4.10. Tras esto, el diseño se puede exportar con extensión *.dae* para ser implementado en el simulador.

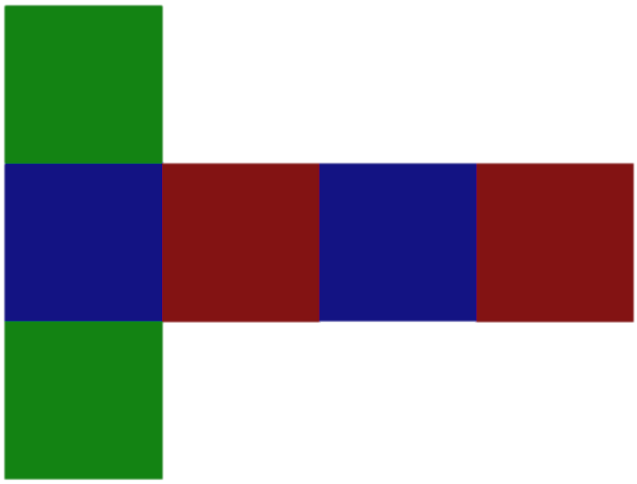

Figura 4.9 Mapa de texturas para el modelo del edificio

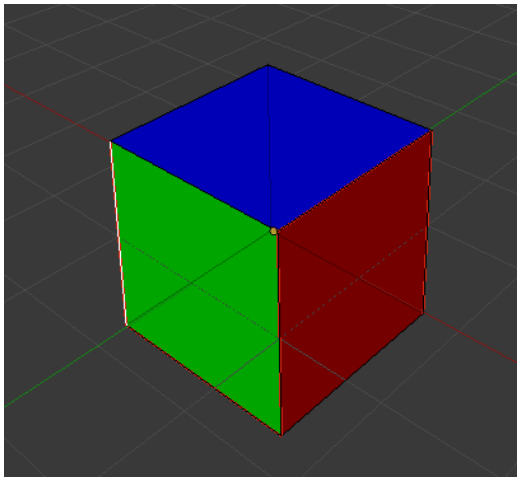

Figura 4.10 Modelo final de los edificios en Blender

Tras reemplazar los bloques sólidos por los nuevos modelos en color, el diseño del entorno de simulación se puede dar por concluido. El resultado final se puede observar en la figura 4.11. Al igual que en el caso de la previsualización, los colores ayudan en gran medida a tener siempre una idea de la orientación dentro del entorno, especialmente dentro de Gazebo donde localizar las referencias de los ejes puede ser más complicado por no estar siempre visibles.

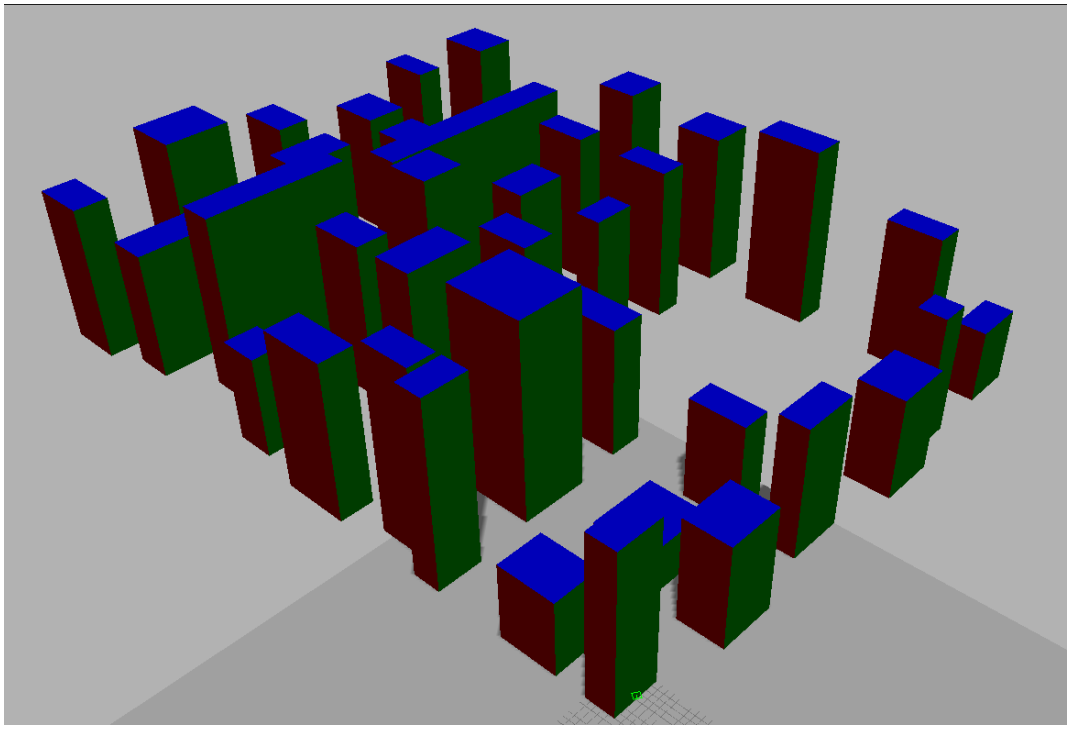

Figura 4.11 Representación final del entorno en Gazebo

4.2.4 Puesta en conjunto de la simulación

Una vez se tienen todas las partes que individualmente componen la simulación, se debe crear una forma de iniciar el conjunto con todos los elementos necesarios en ella. Este objetivo se debe hacer mediante comandos de consola roslaunch. Estos comandos se encargan de ejecutar ficheros de texto que contiene las inicializaciones de ROS a todas las llamadas a la ejecución del resto de elementos necesarios para iniciar la simulación.

La jerarquía y el orden de las llamadas de los ficheros más importantes y de mayor orden en la jerarquía, junto con sus respectivas extensiones se pueden ver representados en la figura 4.12. Esta representación no pretende mostrar la organización completa del programa, ya que la cantidad de archivos es muy superior al mostrado en la figura. Solo se han representado aquellos ficheros que han tenido que ser modificados y por tanto son los más relevantes para el proceso, aunque el número total sea muy superior al mostrado.

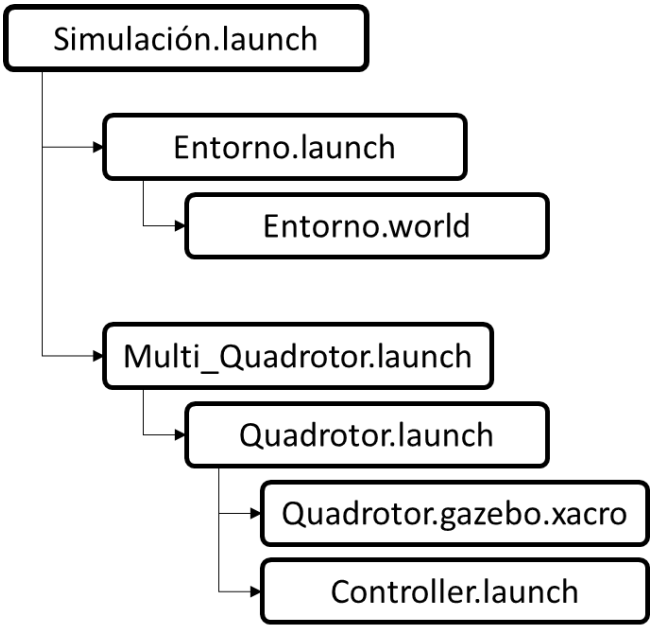

Figura 4.12 Esquema de la estructura de ficheros de la simulación

En la figura se puede ver como en el fichero principal llama tanto al lanzador del entorno como al sistema encargado de generar los drones. A continuación, se explica un poco más en detalle las funciones que desempeñan estos ficheros.

El fichero *Entorno.world* es donde el modelo tridimensional creado en Blender con su mapa de texturas se carga en la simulación. En él se crean tantas instancias como edificios se requiere crear. También se definen los dos parámetros más importantes de cada uno de ellos, su posición y tamaño. Su posición es definida únicamente mediante las coordenadas del centro de masa del edificio y su tamaño se modifica median el escalado en sus tres ejes, haciendo uso de la sentencia scale, *<scale> X Y Z </scale>* donde los valores de X Y Z, gracias a su tamaño original de 1, se pueden asignar directamente las dimensiones en cada eje.

La segunda parte del fichero de inicio es la encargada de realizar las llamadas de ejecución de los drones. Para iniciar un único dron solo es necesario llamar al fichero *Quadrotor.launch*. Éste fichero contiene la inicialización tanto la parte visual y física del modelo, *Quadrotor.gazebo.xacro*, como el controlador interno, *Contoller.lauch*, que inicia todos los nodos y sistemas necesarios para realizar la comunicación y control mediante ROS.

Sin embargo, en el proceso para iniciar varios drones idénticos en la misma simulación existe una complicación adicional. No basta con llamar varias veces a la misma instancia del dron, es necesario crear una subdivisión entre ellos para diferenciar los nodos de comunicación de los distintos drones. Es para esto que se emplea el fichero adicional *Multi_Quadrotor.launch*. Este fichero inicialmente llama tantas veces a la creación del dron como drones intervengan en la simulación, pero con la particularidad de que a cada copia se le asocia un sub espacio de trabajo distinto dentro de ROS. De esta forma se permite que existan múltiples nodos con el mismo nombre y tipo de información pero que están asociados y disponibles únicamente para el dron en el subespacio correspondiente. Esto se puede hacer gracias a la sentencia *<group ns="drone_N"/>* siendo drone_N el nombre asignado al subespacio propio del dron en cuestión, que en este caso se sustituye N por el número de orden arbitrario de cada dron.

Gracias a que todos estos archivos son ficheros de texto con distintas extensiones, es posible automatizar la creación de la mayor parte de ellos en Matlab, prescindiendo así de la edición manual con editores de texto. Dentro del programa principal para la obtención de las trayectorias se ha incluido en su parte final el código responsable de crear de forma automática estos ficheros.

La implementación de esta forma es relativamente sencilla, ya que es durante la ejecución del programa principal de cálculo cuando se generan todos los datos necesarios para crear el entorno. Gracias a esto, solo es cuestión de seleccionar y dar el formato a los datos adecuados para que formen los distintos archivos de lanzamiento de la simulación, que se guardan en el sistema con los formatos adecuados.

Tras su obtención el único paso que es necesario realizar de forma manual es copiarlos dentro del sistema Linux en las carpetas adecuadas. tras actualizar los archivos, cuando se vuelve a ejecutar la simulación los nuevos parámetros aplicados se muestran en la simulación.

4.3 Sistema de control

Una vez se ha calculado la ruta que se desea que recorran los drones y se tiene el entorno en el que se quiere desarrollar, queda por implementar el controlador que lo lleve a cabo. Este controlador es el encargado de enviar los comandos adecuados para que, de forma automática, cada dron siga su propia trayectoria tratando de ajustarse de la forma más precisa posible.

Para diseñarlo, se ha hecho uso de la herramienta Simulink, que es parte del entono de Matlab en el sistema operativo Windows previamente usado. Esto tiene la ventaja adicional de que en principio no es necesario realizar el transvase de los datos previos calculados, ya que las variables de memoria se pueden compartir entre las dos aplicaciones, lo que facilita en gran medida el proceso de desarrollo.

4.3.1 Configuración previa

Para diseñar este controlador, el primer paso es realizar las configuraciones iniciales necesarias. En este caso se trata de establecer la conexión del sistema controlador en Simulink con el simulador en Gazebo como se ha mencionado anteriormente. Esta conexión permite que se comuniquen los dos entornos empleados incluso estando en sistemas operativos distintos. Para ello la conexión se realiza mediante conexiones de red empleando direcciones IP. En este caso, al estar alojados ambos sistemas en el mismo ordenador, se encuentran en la misma red, por lo que solo se tiene que seleccionar la dirección asociada al simulador dentro de la red local, pero podría controlarse remotamente si se tratara de equipos independientes siempre que se tuviera acceso a los equipos implicados.

Una vez se ha establecido la comunicación correctamente entre sistemas, el proceso previo de configuración se puede considerar completado y se puede comenzar a desarrollar el controlador. En Simulink la programación se realiza mediante una combinación de uso de bloques de función y programación clásica con líneas de código. El primer tipo es una programación más visual en la que los distintos bloques realizan transformaciones y operaciones específicas. Las conexiones entre ellos representan los datos que se transmiten unos a otros, por lo que los flujos de información son también más gráficos. Existen gran variedad de bloques de programación que pertenecen a las distintas librerías empleadas en la programación como son ROS, Simulink extras, HDL coder o Robotics system toolbox UAV.

Además de la gran selección de bloques pertenecientes a las librerías empleadas, también se permite crear bloques propios en los que insertar código para crear bloques altamente específicos. De esta forma se pueden crear bloques lo suficientemente únicos como para que no exista una librería que lo contemple o simplemente sea más sencillo crearlo de cero que buscar la librería que contenga la función deseada.

De entre los bloques más importantes y específicos se encuentran los bloques *Subscribe*, *Blank Message*, y el *Publish*, todos pertenecientes a la librería de ROS. Estos permiten la creación y recepción de mensajes desde y hacia el dron, ya que lo dotan de la estructura de datos adecuada para la comunicación además de permitir conectarse a los distintos topics del dron. En esta aplicación concreta, los datos que se quieren recibir son únicamente las coordenadas que describen la posición del dron junto con el ángulo en radianes de su orientación, con lo que su posición exacta queda totalmente definida para el propósito de este proyecto. Los datos enviados al dron por su parte son las 3 velocidades lineales en los ejes cartesianos del propio dron y la velocidad angular para su orientación en el eje Z. [15]

Un ejemplo de la configuración más sencilla que se puede tener en Simulink para controlar un dron de esta forma es la que se muestra en la figura 4.13. En ella se puede observar un controlador completo manual, que mediante el uso de tan solo los 8 botones situados en el lateral izquierdo es capaz de controlar los movimientos en los 3 ejes. Estos botones al ser pulsados establecen una velocidad constante en la dirección y sentido asociados al eje de cada botón. Estos valores se comunican al siguiente bloque, mediante la agrupación en un bus de datos, que proporciona el formato indicado por el bloque *Blank message* en configuración de mensaje tipo *geometry_msgs/Twist*. A la salida de este bloque se conecta al bloque *Publish* que se encarga de transmitirlo directamente al topic que el dron lee para determinar su velocidad, */cmd_vel*.

Haciendo uso de este sistema básico se pueden hacer las primeras comprobaciones de que la transmisión de datos entre los sistemas funciona correctamente, además de comprobar el movimiento del dron y que las interacciones con el entorno responden de forma adecuada.

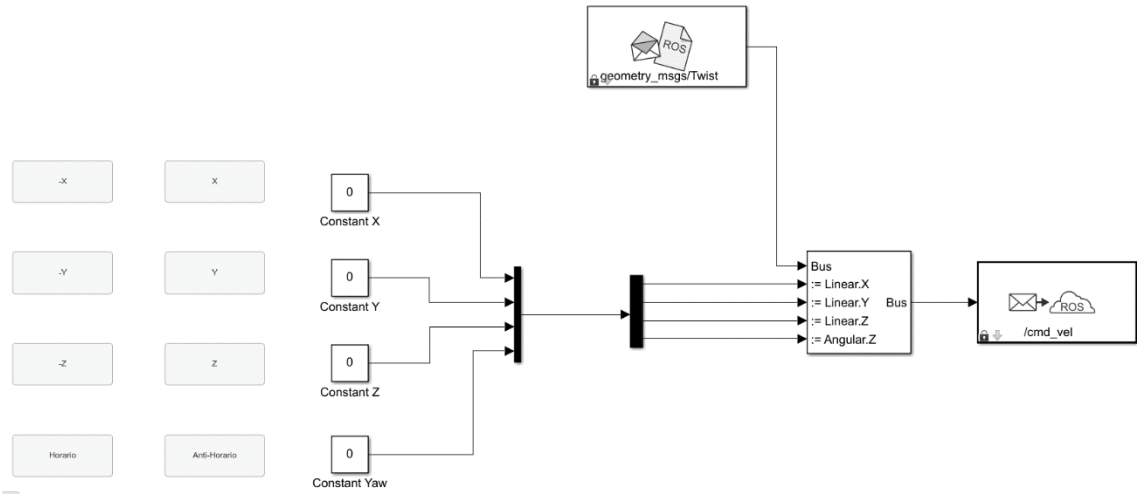

Figura 4.13 Sistema de control manual de drones en Simulink

4.3.2 Desarrollo del controlador

Tras comprobar la capacidad de control manual para asegurar el funcionamiento, se debe reemplazar por un sistema que de forma automática se ajuste a los puntos de la trayectoria previamente calculados. Para ello se ha empleado una aproximación que trata de corregir la posición instantánea del dron con las coordenadas calculadas más cercanas en cada momento. Para su ejecución se ha implementado un controlador PID cuya entrada es la diferencia entre la posición actual y la deseada. La salida de este controlador será un valor que sea de una forma u otra consecuencia de esta diferencia, y será la que se use para determinar la velocidad. Cada eje coordenado, además de la responsable de la rotación del dron tendrán su propio controlador, de esta forma se permite un ajuste más preciso dependiendo de sus necesidades específicas.

Durante el desplazamiento del dron a lo largo de la trayectoria, éste mantiene una orientación fija durante todo el recorrido. En este proyecto no existe ninguna razón por la que debería variar ya que no requiere de orientación para cámaras o algún otro sensor. Por tanto, el dron mantiene alineados sus ejes con los del entorno, simplificando de esta forma el proceso y no haciendo necesaria una transformación entre sistemas de referencia. Como consecuencia, los desplazamientos en un determinado eje del dron siempre se corresponden con los mismos desplazamientos en el entorno.

No obstante, si en futuras implementaciones del proyecto se requiriera una orientación variable en función del instante de la simulación, no sería difícil de implementar. Bastaría con incluir un bloque que transforme las velocidades referidas al sistema de coordenadas del mundo a las velocidades que el dron es capaz de procesar, mediante simple descomposición de los vectores en cada dirección. Esta es una de las ventajas que presenta tener una programación por bloques, que es muy fácilmente modificable y adaptable a nuevas necesidades.

Realizando esta implementación con el eje Z constante, podría parecer que no es necesario mantener el control sobre la orientación del eje por no requerirse cambios en él. Sin embargo, debido a las aceleraciones que sufre el dron en distintas direcciones, se puede comprobar que la orientación sufre una deriva aleatoria con el tiempo, lo que hace necesario una pequeña, pero continua corrección en el tiempo. De esta forma, el controlador de la orientación o Yaw se puede establecer para que corrija de forma suave estas pequeñas variaciones aleatorias manteniéndolo siempre orientado correctamente.

El último aspecto importante que hay que considerar es que obtener las coordenadas objetivo en cada instante de tiempo no es un aspecto trivial. Las coordenadas instantáneas del dron se pueden leer directamente de su posición en el simulador, pero determinar cuál es la posición que se debe perseguir en cada momento requiere de manipulación adicional de los datos para obtenerla.

Para determinar el punto concreto de la trayectoria que se debe alcanzar se debe emplear un algoritmo que busque de entre todos los puntos de la trayectoria y aquellos que se encuentran en posiciones intermedias, aquel que se localice a una distancia dada del dron y que alcanzarlo permita avanzar hacia el último punto de la trayectoria. Un algoritmo que realiza esta tarea es el denominado PurePursuit.

Para la implementación de un PurePursuit en Simulink se disponen de numerosas opciones, pero el bloque finalmente empleado es el *UAV Waypoint follower* de la libreria *Robotics System* Toolbox UAV library, configurado en su modo para manejo de drones. Como se puede ver en la figura 4.14, el bloque tiene tres entradas y cuatro salidas.

Las entradas del bloque se corresponden de arriba abajo con la posición instantánea del dron, el vector que contiene todas las coordenadas del su trayectoria y la distancia a la que el algoritmo debe calcular el punto hacia el que se debe dirigir, el *Look ahead distance*, también referido como radio de búsqueda.

De sus cuatro salidas únicamente se ha empleado la primera de ellas. Se trata de su output principal, donde se puede obtener el resultado de la búsqueda de coordenadas que cumplen con los requerimientos establecidos. La segunda de las salidas al no trabajar con requerimientos de orientación se puede ignorar y la tercera y cuarta son enfocadas al trabajo con UAVs de ala fija, el otro tipo de vehículos para los que permite realizar los cálculos esta librería.

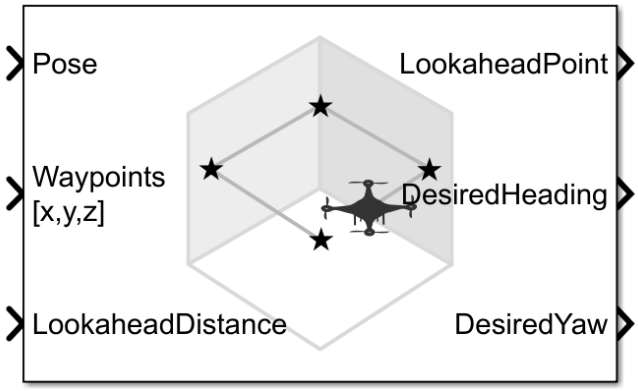

Figura 4.14 Bloque UAV Waypoint follower en el entorno de Simulink

El Look head distance es el parámetro modificable más importante a la hora de emplear este PurePursuit, ya que el resultado obtenido y su comportamiento varía en gran medida dependiendo del valor empleado. Valores bajos de este parámetro provocan que el dron se desplace rápidamente hacia la trayectoria, pero como contrapartida puede oscilar en torno a él en exceso. Por el contrario, valores elevados desplazan el robot hacia la trayectoria de forma más lenta, lo que puede provocar que se aleje de la trayectoria deseada pudiendo llegar a saltarse puntos por completo. Una representación esquemática de ambos comportamientos se puede ver en la figura 4.15, donde a la izquierda se tiene el resultado con un valor bajo y a la derecha con uno alto. [16]

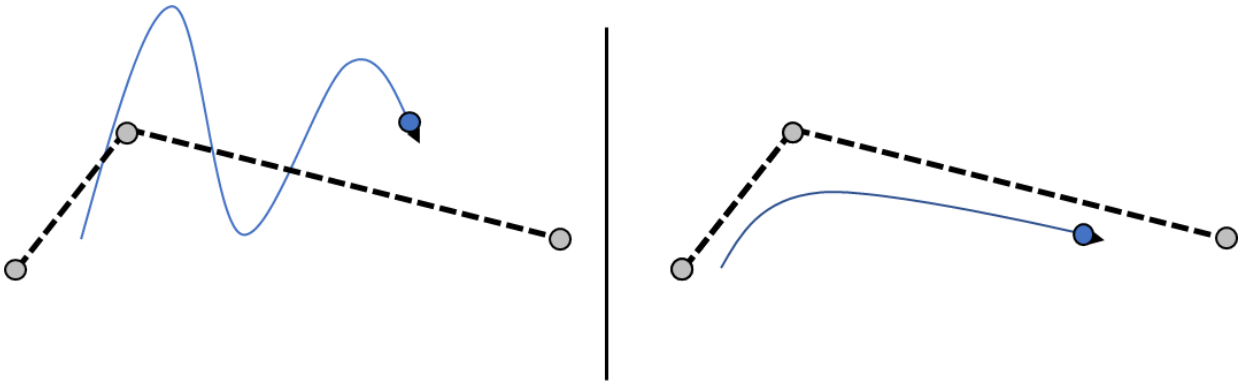

Figura 4.15 Comparativa de los resultados para distintos valores del parámetro Look ahead distance en un sistema de PurePursuit

Por este motivo es necesario encontrar un valor adecuado para el sistema concreto de dron y controlador empleados. Para lo que la única manera de llegar a ese valor optimo es mediante prueba y error, examinando los resultados después de cada modificación para obtener el comportamiento deseado. Este proceso junto con sus valores seleccionados se comentará en detalle en el *Capítulo 5* de los resultados obtenidos.

En la figura 4.16 se puede ver el resultado final del sistema de control para un único dron. En la parte central se puede ver el bloque PurePursuit descrito. A su izquierda se encuentra el sistema de adquisición de datos en tiempo real del dron. En la parte derecha del PurePursuit se encuentra el sistema de control para la velocidad y, por último, el bloque más alejado es el sistema encargado de enviar los datos de velocidad hacia el dron.

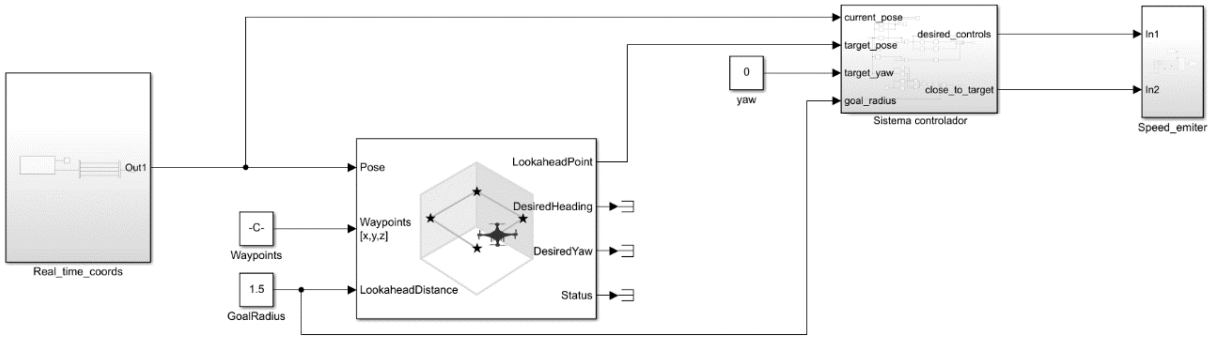

Figura 4.16 Vista general del sistema controlador de Simulink

Simulink además de permitir colocar bloques con funciones predefinidas o diseñarlos desde cero con código, también permite la agrupación de combinaciones de todo ellos. Estas agrupaciones permiten crear estructuras más compactas que no solo están mejor organizadas en el espacio de trabajo, si no que resultan en estructuras muy potentes formadas únicamente por agrupaciones de operadores más simples. Este es el caso de los bloques visibles en la imagen anterior, que, aunque aparentan ser simples bloques con algunas interconexiones, sus componentes internos tienen una gran complejidad en especial el sistema controlador. De todos ellos se va a tratar de dar una idea de los procesos internos que llevan a cabo y se detallan a continuación.

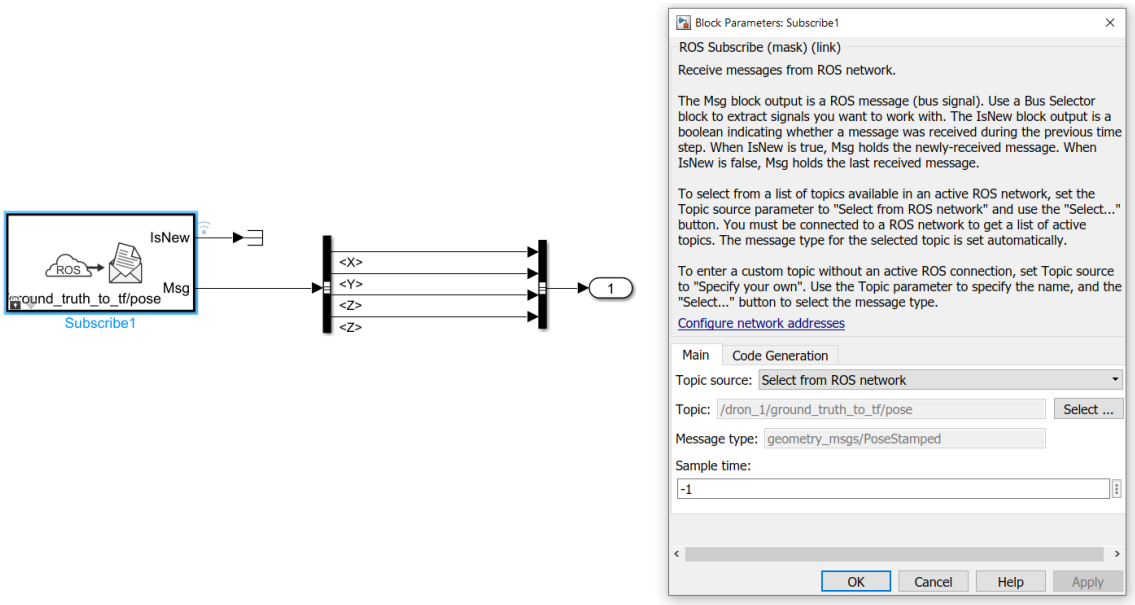

Figura 4.17 Detalle del conjunto receptor del controlador en Simulink

El contenido del bloque de recepción de la información se puede ver en la figura 4.17. En él se aprecia la aparición del bloque Subscriber. Se ha configurado para que reciba el topic */ground_truth_to_tf/pose* del dron 1, motivo por el cual tiene el prefijo /dron_1 de su sub espacio asociado donde este y solo este dron publica sus actualizaciones de estado. Tras conectarse al topic se debe seleccionar la información necesaria, en este caso solo son necesarias las 3 coordenadas cartesianas y la orientación en el eje Z. Mediante la creación de un bus de datos se define el orden y se compacta de forma que la salida del conjunto del bloque sea un único bus de datos que simplifica en gran medida la comprensión del conjunto.

Por otra parte, el bloque controlador, el más complejo de los que se han diseñado, contiene todos los sistemas que se encargan propiamente del cálculo de los movimientos necesarios. Su configuración interna se muestra en la imagen 4.18.

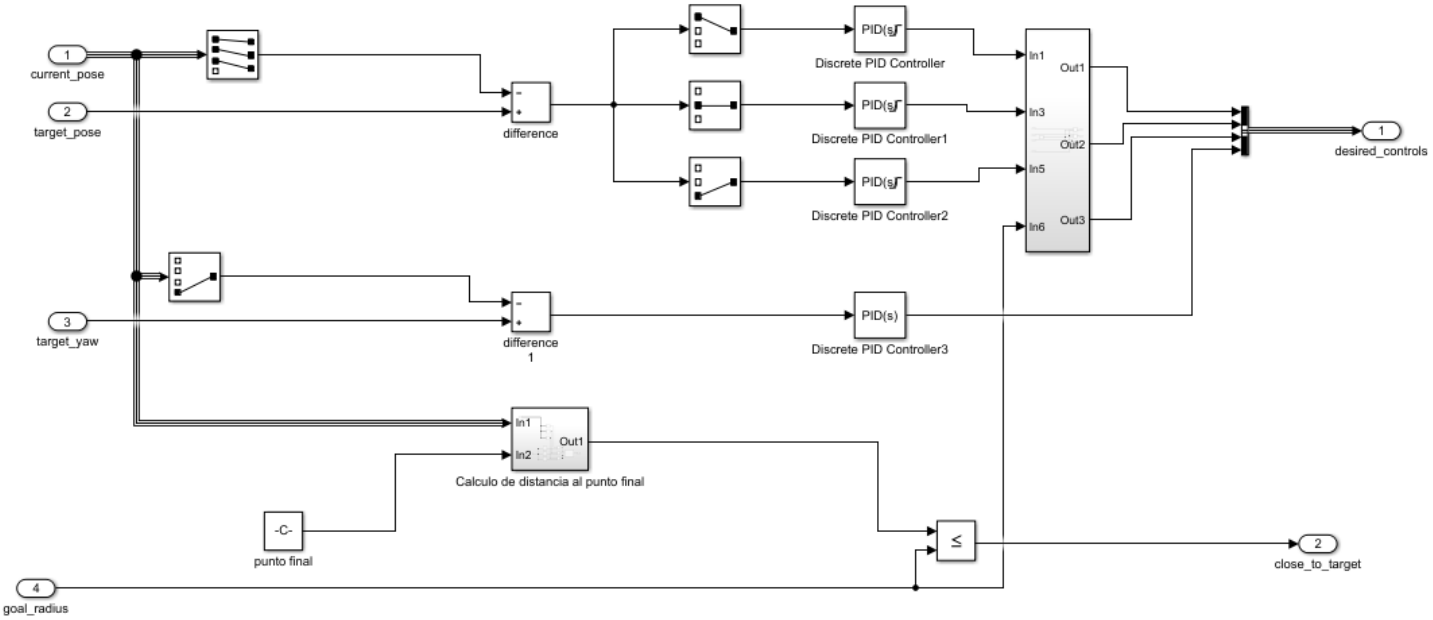

Figura 4.18 Detalle del conjunto principal de control en Simulink

Como se podía ver en la imagen general del programa, este bloque tiene cuatro entradas y dos salidas. Las entradas en orden descendente se corresponden con la posición actual, el punto objetivo obtenido en el PurePursuit, el ángulo de Yaw objetivo (fijo en 0 radianes para este proyecto) y el valor de Look ahead distance empleado en el PurePursuit.

En la parte superior y central de este conjunto se encuentran los bloques destinados al cálculo de las velocidades. Se seleccionan las 3 coordenadas cartesianas actuales que se restan a las análogas de las coordenadas objetivo. El resultado de cada una de las 3 restas se hace pasar por sus correspondientes controladores PID, y posteriormente por el bloque que determina la propia velocidad que el dron debe tomar en cada eje, la cual se determina en función de las salidas de los PID y del valor de velocidad máximo estipulado mediante variables de memoria. La salida se combina en un bus de datos con el análogo a las velocidades lineales en los ejes, la componente angular del eje z. El bus resultante contiene las cuatro componentes y es la primera de las dos salidas del conjunto del controlador, la consigna de velocidad del dron.

Debido a que el sistema de PurePursuit no emite un output distinto al alcanzar la coordenada final de una trayectoria, y que al llegar a ese punto los outputs del sistema carecen de sentido lógico para la simulación, es necesario tener un sistema que permita determinar cuándo se ha completado la ruta planificada dejando de enviar ordenes de movimiento al dron.

Para ello se tiene la parte inferior de este bloque de control. Se puede observar cómo se tiene un bloque que determina la distancia que existe desde la posición actual hasta el último punto del conjunto de la trayectoria. Cuando el valor de esta distancia es menor que el valor arbitrario seleccionado, en este caso la misma distancia que el radio de búsqueda del PurePursuit, la salida será un valor booleano verdadero, mientras que en el resto del recorrido este valor será falso.

Estas son las dos únicas opciones que tiene la segunda salida del sistema controlador. Ambas se conectan directamente al módulo de *speed_emiter*, la tercera agrupación de bloques que se encarga de transmitir los datos que se acaban de procesar al sistema de dirección del dron. Este bloque se muestra en la figura 4.19 y se detalla a continuación.

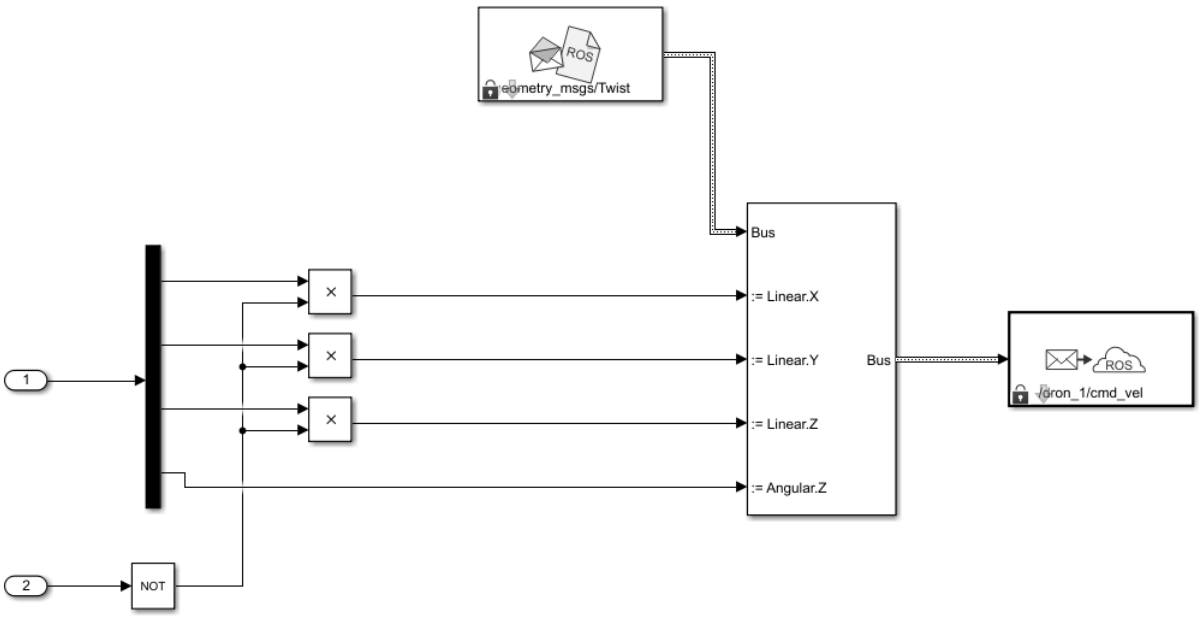

Figura 4.19 Detalle del conjunto emisor del controlador en Simulink

Se puede apreciar en la parte izquierda como el bus de las velocidades, marcado con la entrada 1, se divide en sus cuatro componentes para que mediante el bloque de creación de buses y el bloque el *blank message* dotar al bus de salida de la estructura tipo Twist necesaria para poder transmitir la información al dron. Este envió se completa al conectar el bus al bloque de ROS, *Publish*, que permite seleccionar a cuál de ellos enviar esta información. En este caso por ser el dron 1 y tratarse de información de velocidades es el topic */dron1/cmd_vel*, con lo que termina el viaje de la información por el controlador y se envía directamente al dron correspondiente.

La entrada 2 de este bloque es la señal booleana de proximidad al objetivo. Mediante la operación de negación con un bloque NOT y el uso de un bloque AND en cada componente de la otra entrada, se puede conseguir que solo se transmita las consignas de velocidad si no se ha llegado al entorno del objetivo del dron, evitando así que se mueva una vez que el dron se encuentre próximo al objetivo.

Este último conjunto de bloques no requiere de salidas de datos dentro del programa, ya que el recorrido final de la información se realiza fuera del programa, dentro del propio simulador. Al hacer uso del bloque *Publish*, no es necesaria ninguna salida adicional de buses de datos.

El ajuste de los parámetros mencionados en el control se realiza en el *CAPÍTULO 5: Resultados del proyecto*, al igual que el parámetro de *Radio de búsqueda*. En el siguiente capítulo mediante iteraciones experimentales se tratará de buscar los valores que consiguen un mejor comportamiento del sistema en las simulaciones finales.

Este es en resumen un programa que es capaz de realizar el control en tiempo real de un dron. El último paso que es necesario es realizar para poder dar inicio a las simulaciones es poder controlar múltiples drones simultáneamente. A diferencia del procedimiento en Gazebo, en Simulink si se puede copiar el controlador completo y realizar varias copias de él. La única modificación que se requiere además de los evidentes cambios en los sets de coordenadas asociados, es cambiar los topics asociados a la emisión y recepción de los datos. Se debe buscar el topic equivalente para cada función, pero usando el sub espacio adecuado para cada dron. De esta forma en lugar de buscar el topic de emisión de velocidad */cmd_vel*, se debe seleccionar antes el sub espacio correspondiente: */dron_N/cmd_vel*, siendo N el numero asociado al dron en cuestión, de esta forma es bastante sencillo añadir múltiples drones al sistema de control una vez se han realizado los ajustes previos en el entorno de Gazebo.

Gracias a la posibilidad de agrupar grupos de bloques complejos en otros más simples en apariencia, el programa completo con tres controladores tiene el aspecto que muestra la figura 4.20. Se puede ver como todo el programa queda simplificado de forma que cada controlador aparenta ser un único bloque en el programa final sin entradas ni salidas de buses de datos, en el que se intuye la forma general mostrada en la imagen 4.16. De esta forma, añadir o quitar controladores de nuevos drones resulta bastante sencillo además de facilitar la organización en caso de necesitar realizar modificaciones.

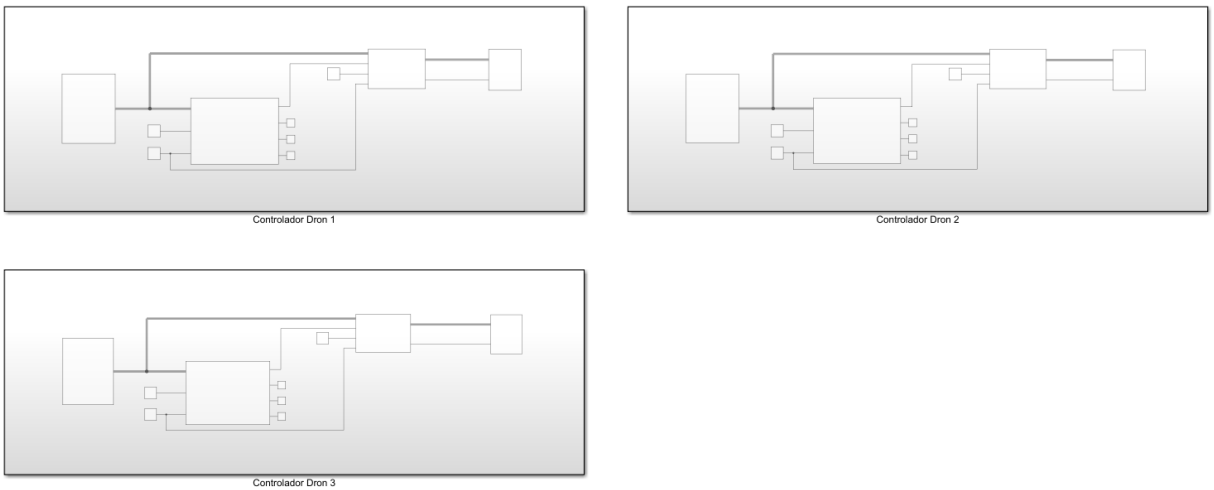

Figura 4.20 Visión completa del sistema en Simulink para el control de 3 drones

4.3.3 Monitorización de los resultados

Una parte muy importante en este tipo de proyectos de es la monitorización y análisis de resultados. Para ello, se debe implementar en algún punto del sistema los elementos que permitan visualizar los datos más importantes, además de permitir guardarlos para poder realizar un análisis posterior.

En este proyecto el lugar más adecuado para su implementación es dentro del propio controlador principal en Simulink, ya que tiene acceso a todos los datos del proyecto. Los referidos a los drones, de carácter experimental, son recibidos mediante los distintos topics, mientras que los que se refieren a la estructura del entorno y la planificación de trayectorias previa se pueden leer directamente de las variables de memoria del programa de cálculo ejecutado en Matlab.

Para leer los datos en tiempo real de posición de los drones se han empleado bloques que permiten adquirir y graficar los distintos valores. Estos bloques son el *Scope* y el *X-Y Graph*, aunque por la mayor capacidad de personalización en su mayoría se ha tratado con los bloques *Scope*. La función de este componente es graficar los valores de sus inputs a lo largo del tiempo. Su aplicación más directa para la que se ha empleado es la de representar la posición instantánea de cada dron. Para ello se puede modificar para que acepte los 4 inputs que definen la posición en cada momento, y muestre su evolución en un periodo de tiempo determinado, generalmente el periodo de toda una simulación.

El otro elemento importante en esta fase de recogida de datos es el bloque *To Workspace*. Este bloque permite, al asociarlo a un bus de datos, guardar toda la información entrante en variables de memoria de Matlab. De esta manera se pueden crear gráficas a conveniencia tras finalizar el proceso de simulación en Gazebo.

Estos elementos se introducen en el bloque del sistema de control principal como se puede ver en la figura 4.21 donde se encuentran resaltados todos los elementos de observación añadidos a los que se mostraban en la figura 4.18.

Con esto se concluyen los aspectos más importantes en lo referente a la programación e implementación del proyecto. En el siguiente capítulo se mostrará de forma detallada los resultados que se han conseguido alcanzar durante el proceso final de simulación de misiones completas, así como las modificaciones que se han visto necesarias realizar en el proceso.

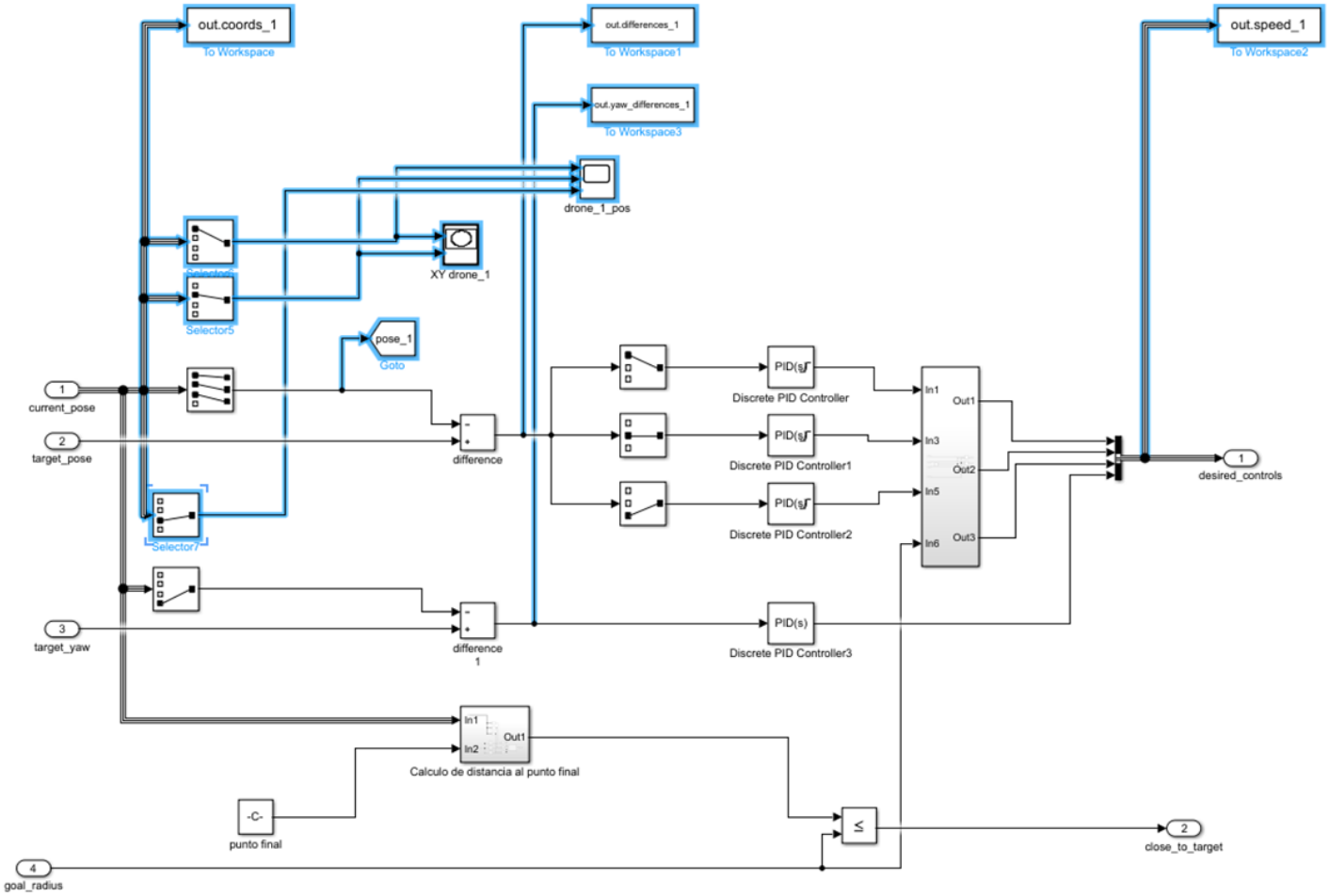

Figura 4.21 Detalle del conjunto del controlador principal en Simulink con sistemas adicionales de adquisición de datos

CAPÍTULO 5: Resultados del proyecto

En este quinto capítulo se pretende mostrar las capacidades del conjunto del proyecto mediante unas simulaciones finales. Estas se han realizado con el objetivo de reflejar distintas situaciones que abarquen un gran abanico de misiones que podrían simularse haciendo uso del sistema final. Para ello, aunque en principio no debería ser necesario implementar o modificar ninguna parte sustancial a la programación, sí que se van a añadir elementos que, sin ser estrictamente necesarios, mejoran el funcionamiento del conjunto. Además de este motivo, también se añadirán algunos sistemas que permitan visualizar nuevos datos de las simulaciones que permitan realizar un análisis más detallado a posteriori y una documentación más adecuada para esta memoria.

El aspecto que se ha tenido que concretar durante la elaboración de este capítulo es la definición de los parámetros que rigen el comportamiento de los drones y sus sistemas asociados de control. Estos, debido a su dependencia de la misión y el entorno concretos, se han obtenido mediante iteraciones en simulaciones. Se han ajustado en la medida que se ha comprobado adecuadas para la misión en concreto, demostrando la facilidad con la que estos se pueden adaptar a los requerimientos de la simulación. Teniendo el objetivo adicional de obtener unos valores genéricos que tengan buenos resultados sin gran dependencia de la misión en concreto.

Se han realizado tres misiones distintas en el entorno urbano definido. En estas misiones se han empleado tres drones en cada una para facilitar la labor de análisis posterior de los datos, siendo un número suficiente para comprobar las interacciones que ocurren entre ellos, sin ser un demasiado elevado como para que no permita una representación clara de los resultados.

5.1 Primera misión. Ajustes generales de la parametrización

El objetivo principal de esta misión, ha sido encontrar valores adecuados de los parámetros empleados en los controladores de los drones que sirvan como base para el resto de simulaciones. Los parámetros modificados son tres y sus descripciones más detalladas se encuentran en el sub apartado *4.3.2 Desarrollo del controlador*. Estos parámetros son la distancia de búsqueda en el PurePursuit, las velocidades máximas de los drones y los valores de los controladores PID.

Aunque estos valores se pueden ajustar de forma individual a cada dron, el objetivo de crear un simulador valido para drones de forma general ha hecho que se emplee el mismo valor para un determinado parámetro en todos los drones que participan siempre que fuera posible, aunque siempre se mantiene abierta la posibilidad de modificación individual.

Es por este motivo que se ha tenido un especial cuidado al elegir las trayectorias simuladas. Esto se ha hecho mediante el ajuste de los puntos de inicio y final. Con ello se ha buscado que los paths calculados mediante Fast Marching produjeran un abanico lo más diverso posible para comprobar un mayor rango de situaciones.

Las trayectorias que se han planificado para esta primera misión se muestran en la figura 5.1. Estas son las mismas que se podían ver como ejemplo durante la descripción del sistema, pero en este caso se han representado los edificios en gris para asignar los colores RGB a las trayectorias de cada dron por orden y mantener así una representación más clara y focalizada para esta memoria.

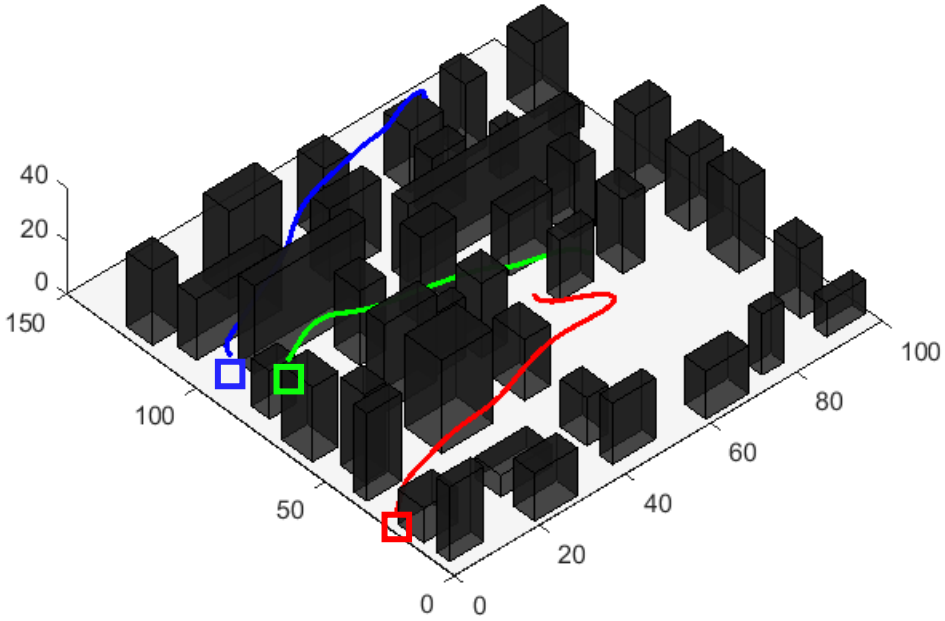

Figura 5.1 Representación tridimensional de la primera misión

Además de esta representación tridimensional se ha hecho necesaria la representación de las trayectorias en gráficas de dos ejes. De esta forma la visualización de los datos es más sencilla y no requiere de la manipulación de la orientación de las gráficas tridimensionales que solo permite Matlab y que en esta memoria resulta imposible. De esta forma en la figura 5.2 se muestra en primer lugar las trayectorias proyectadas sobre el plano X-Y, y en la siguiente imagen, la figura 5.3, el perfil de alturas frente a la longitud de cada trayectoria. El eje X de esta última representación muestra en el número de posiciones calculadas por el algoritmo, pero se puede interpretar como la longitud recorrida por el dron.

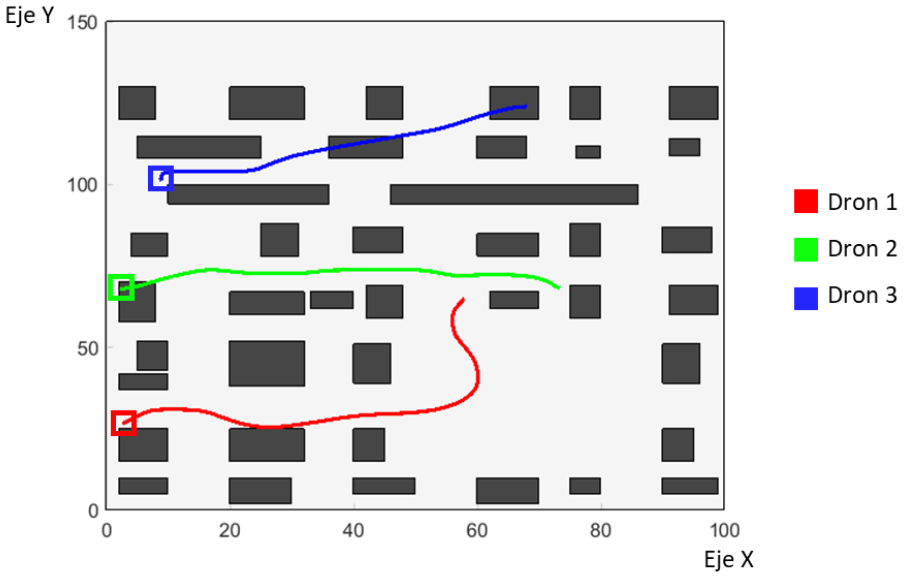

Figura 5.2 Representación en el plano X-Y de la primera misión

En ambas figuras, y en el resto de figuras de este capítulo que lo requerían, se ha marcado con recuadros los puntos de inicio para poder identificar la dirección del desarrollo que sufre cada dron durante las simulaciones. Es este caso los tres se desplazan aproximadamente en dirección paralela al eje X y en su sentido positivo.

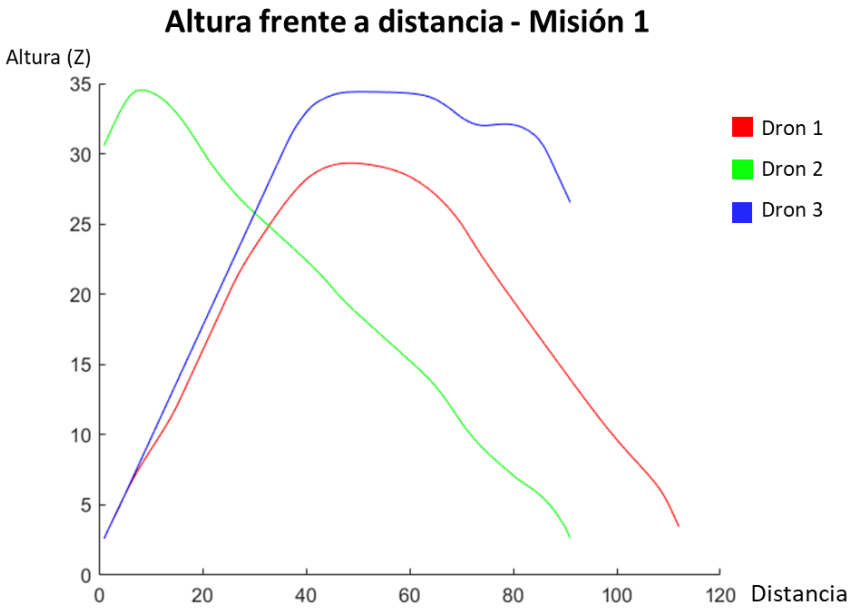

Figura 5.3 Representación de la evolución de la altura en la primera misión

Se puede ver como la trayectoria del dron 1 representada en rojo es la que tiene un perfil de alturas más cambiante, comenzando y terminando desde el plano del suelo, subiendo hasta casi los 35 metros en la parte central de su recorrido. Además, se trata de la más larga de las 3 que intervienen en esta primera misión. El dron 2 en color verde en cambio se ve como comienza sobre un edificio y termina en el suelo. Este es justo el comportamiento opuesto al del dron azul, el tercero de la simulación.

Estas son las trayectorias planificadas mediante Fast Marching. Ahora se va a mostrar cómo se recorren por los drones de la simulación. Para ello hay que seleccionar los valores de los parámetros anteriormente mencionados. Para comenzar se van a seleccionar unos valores arbitrariamente bajos para comprobar el correcto funcionamiento del sistema. Estos van a ser una velocidad de $1\frac{m}{s}$, un Radio de búsqueda de 1 metro y un controlador simple con únicamente parte proporcional ajustada a 1. Estos valores por simplicidad se denotarán como los del Caso 1.

Los resultados obtenidos de esta primera simulación se han representado en gráficas tridimensionales en las que se muestran las dos trayectorias de interés. En primer lugar, las teóricas mostradas en las imágenes anteriores, representadas en color verde. La otra trayectoria que se representa de forma conjunta es la experimental obtenida durante la simulación. Esta se ha representado en color rojo y en discontinuo para diferenciarlas lo mejor posible.

El resultado de esta primera aproximación con los parámetros seleccionados se muestra en las siguientes 3 figuras. Éstas no se han representado con una escala común a todos los ejes. El motivo es poder ampliar lo más posible el espacio de la representación para que se puedan visualizar los puntos en los que las trayectorias difieran de una forma más fácil, aunque como contrapartida pueden parecer algo distorsionadas respecto a las originales de la figura 5.1.

En estas figuras se pueden apreciar como ambas trazas están completamente superpuestas, con la única excepción de la primera parte del trayecto, correspondiente con el despegue. Esto se debe a que la trayectoria calculada no comienza directamente en contacto con el suelo, y por tratarse de una simulación, no se puede comenzar con el dron en esa posición alejada de la superficie, por lo que al inicio de todas las trayectorias se puede observar un inicio en rojo discontinuo.

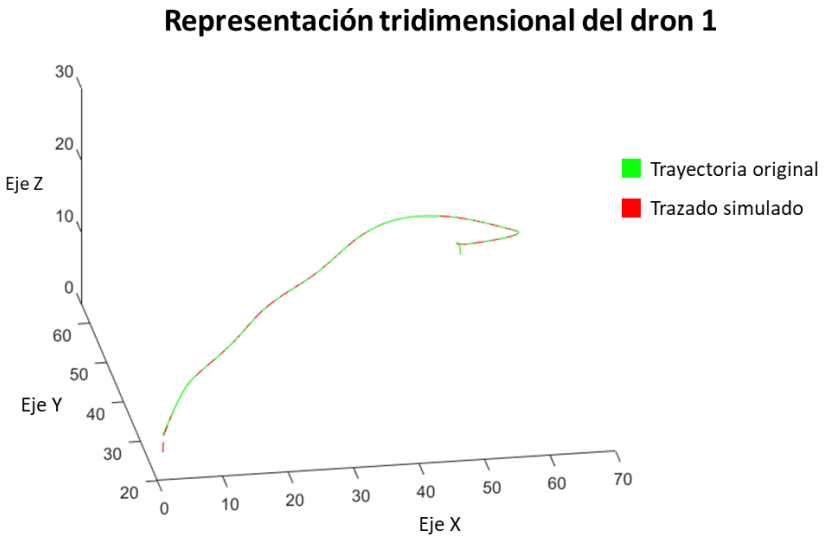

Figura 5.4 Comparativa tridimensional del dron 1. Misión 1 Caso 1

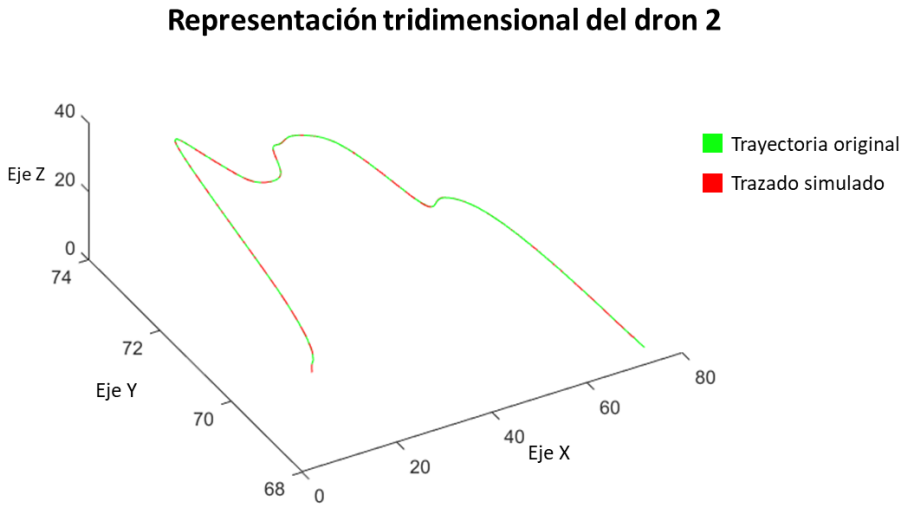

Figura 5.5 Comparativa tridimensional del dron 2. Misión 1 Caso 1

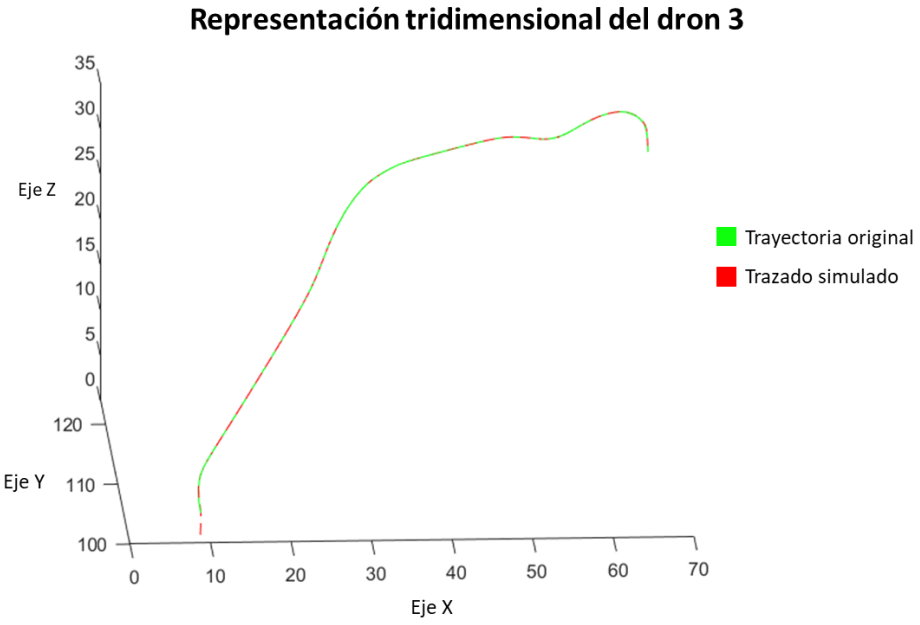

Figura 5.6 Comparativa tridimensional del dron 3. Misión 1 Caso 1

Algo similar ocurre en los finales de trayectoria, donde el dron en la simulación no llega a completar hasta el último punto, esto es debido a que el sistema de detección de final tiene un rango, y por tanto detiene al dron una distancia, que aunque pequeña, se puede apreciar cómo es anterior al último punto de la trayectoria planificada.

En cuanto a la precisión con la que la trayectoria simulada se acerca a la planificada, es muy elevada, sin poder apreciarse ninguna desviación en la totalidad del recorrido. Esto era algo previsible hasta cierto punto ya que se está tratando con velocidades bajas, pero era necesario establecer un punto de partida desde el que determinar los valores definitivos.

Aunque este proyecto no tiene un objetivo concreto respecto al comportamiento de los drones, como criterio para ajustar los parámetros se va a emplear la precisión con la que se ajusta el dron a la trayectoria original y el tiempo empleado. De esta forma el objetivo principal es obtener simulaciones rápidas, pero sin comprometer en exceso el trazado de la trayectoria, y con ello la estabilidad del sistema.

De esta forma, se va a proceder a aumentar de forma progresiva la velocidad para reducir los tiempos empleados para recorrer las trayectorias, ya que esta primera simulación asciende hasta los casi 170 segundos.

Este ajuste, sin embargo, no se puede hacerse de forma independiente a cada parámetro. La velocidad está estrechamente ligada al radio de búsqueda. En la figura 5.7 se puede ver el resultado de aumentar la velocidad a $3\frac{m}{s}$ manteniendo el resto de parámetros igual, lo que se referirá como Caso 2.

En esta imagen se muestra mediante el otro tipo de representación implementada, la obtenida con los bloques *Scope*. En ella se puede ver la evolución a lo largo del tiempo de cada coordenada, nuevamente usando el sistema RGB estándar. Esta es una representación muy útil ya que se puede visualizar al mismo tiempo que se desarrolla la simulación, y permite monitorizar el estado actual de la misma.

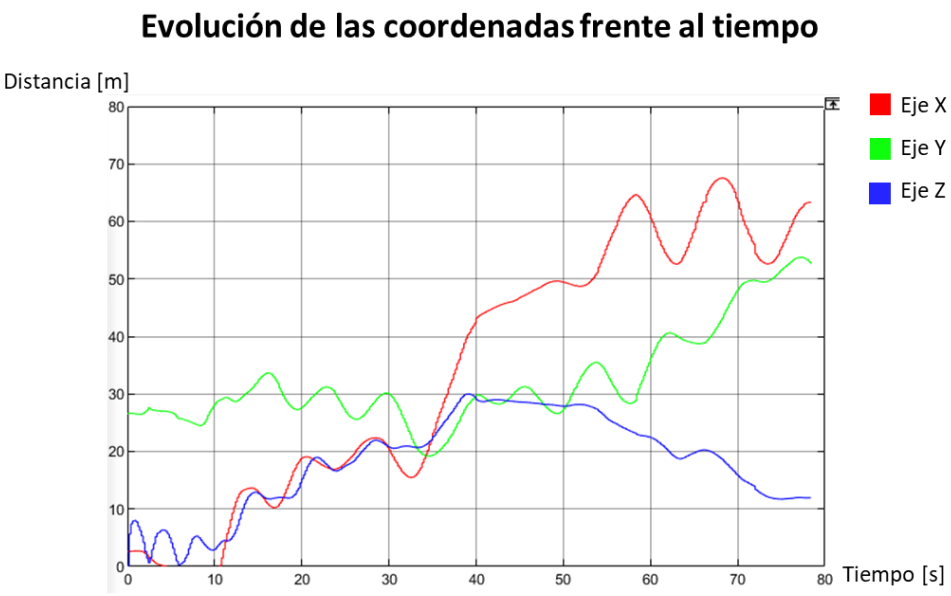

Figura 5.7 Evolución de las coordenadas frente al tiempo. Misión 1, Dron 1 Caso 2

En este caso se trata de los valores obtenidos durante la simulación para el dron 1 realizando la misma trayectoria que el caso anterior. Se puede observar las constantes oscilaciones que sufren todas las coordenadas. Este comportamiento errático es el descrito en la definición del radio de búsqueda y nos indica que su valor es demasiado pequeño. Al aumentar la velocidad, el punto de referencia debe estar más alejado dado que el tiempo que se tarda en alcanzarlo es menor, para permitir un trazado uniforme y sin tantas oscilaciones.

La representación de la trayectoria planificada frente a la simulada de este primer dron se puede ver en la figura 5.8. En ella se puede ver como la representación en el plano X-Y de la trayectoria simulada tiene oscilaciones en torno a la trayectoria deseada, en verde, durante toda la simulación, lo que confirma la suposición obtenida con la figura anterior.

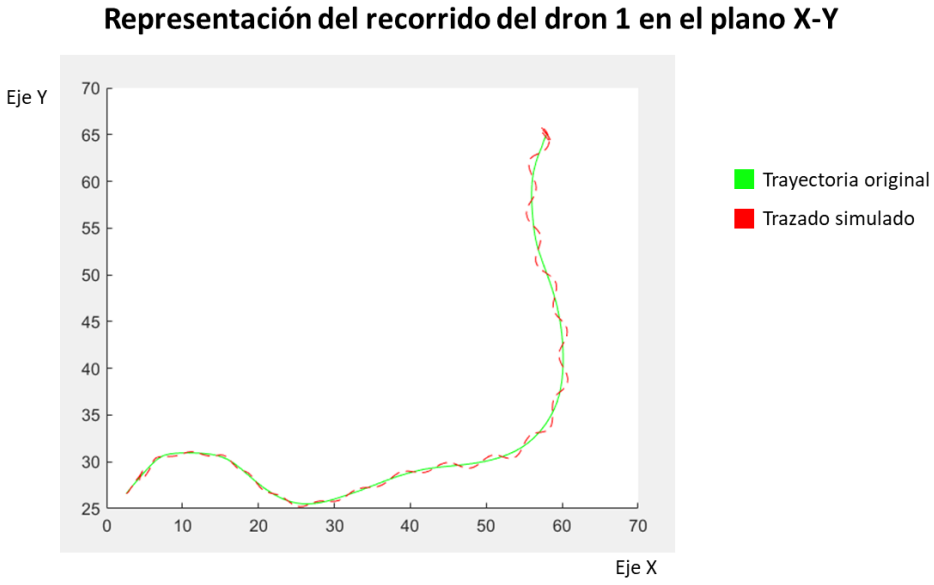

Figura 5.8 Representación del recorrido errático del dron 1. Misión 1, Caso 2

La figura 5.7 anterior del desarrollo en el tiempo se puede contrastar con la figura 5.9 donde se muestra esta misma representación para el dron 1, pero usando la velocidad de 1 $\frac{m}{s}$, la configuración del caso 1. Esta figura, junto con las dos siguientes, la 5.10 y la 5.11, que se corresponden con los recorridos de los drones 2 y 3 en este mismo Caso 1, se puede usar como referencia para esta primera misión, ya que permite tener una idea de la forma que deben tener las gráficas si los drones recorren las trayectorias de forma adecuada.

Se puede apreciar como la evolución del primer dron tiene las mismas tendencias, aunque con continuas oscilaciones. Eso quiere decir que se está recorriendo el trayecto marcado, pero con un recorrido mucho más sinuoso debido a un seguimiento inapropiado del PurePursuit.

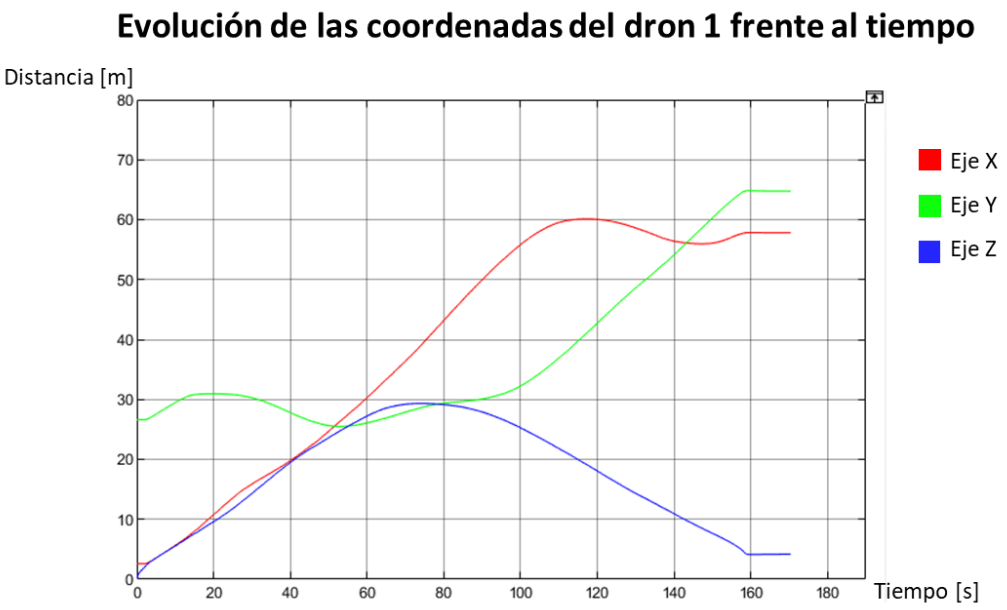

Figura 5.9 Representación de la evolución de las coordenadas del dron 1. Misión 1 Caso 1

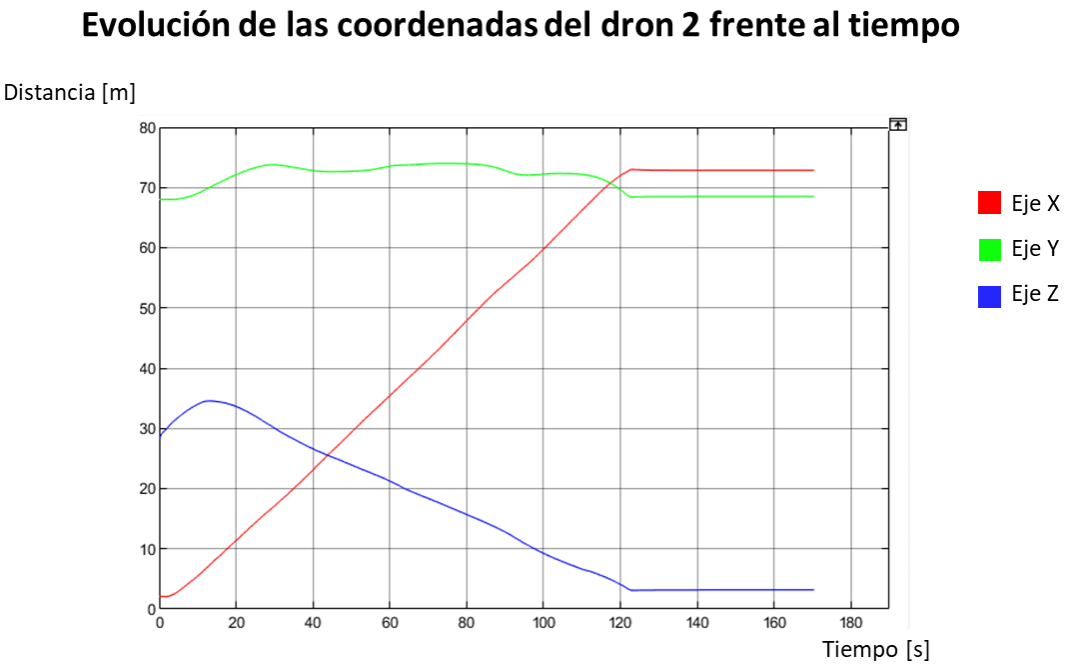

Figura 5.10 Representación de la evolución de las coordenadas del dron 2. Misión 1 Caso 1

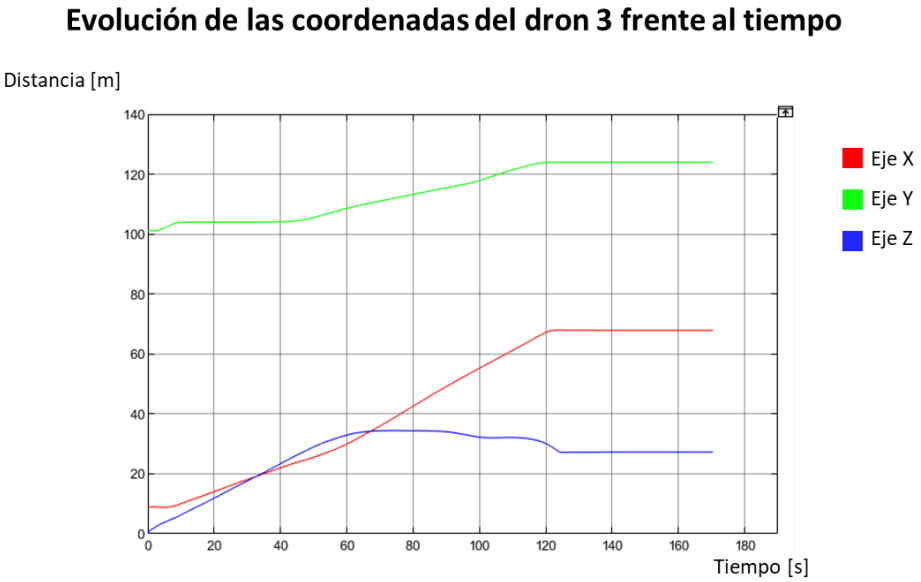

Figura 5.11 Representación de la evolución de las coordenadas del dron 3. Misión 1 Caso 1

Otra utilidad de esta representación es que se puede visualizar la llegada al final del trayecto desde el entorno Simulink en tiempo real. Cuando esto ocurre, los valores de las coordenadas se estancan y apenas cambian, manteniéndose constantes para el resto de la simulación. Sus casi imperceptibles variaciones se deben a que el dron, aunque ha llegado al destino, se encuentra aún sobrevolándolo y puede tener pequeñas desviaciones en la posición debido a ello.

Para esta primera modificación de la velocidad a 3 $\frac{m}{s}$ se puede observar cómo al incrementar el radio de búsqueda a 3.5 metros mejora notablemente el recorrido ejecutado por los drones como se puede ver en las figuras 5.12 a 5.14 en las que se emplea esta nueva configuración denotada como Caso 3. En ellas se puede ver la representación de las trayectorias planificadas y las recorridas en el plano X-Y, que nuevamente no tienen ambos ejes a escala, sino que se ajustan para mostrar lo más ampliado posible los recorridos. Ya no se trata de un recorrido exacto punto a punto como en el Caso 1, pero es algo esperado debido al aumento de las velocidades. Además, estas separaciones se pueden considerar mínimas en la mayoría de puntos de la trayectoria.

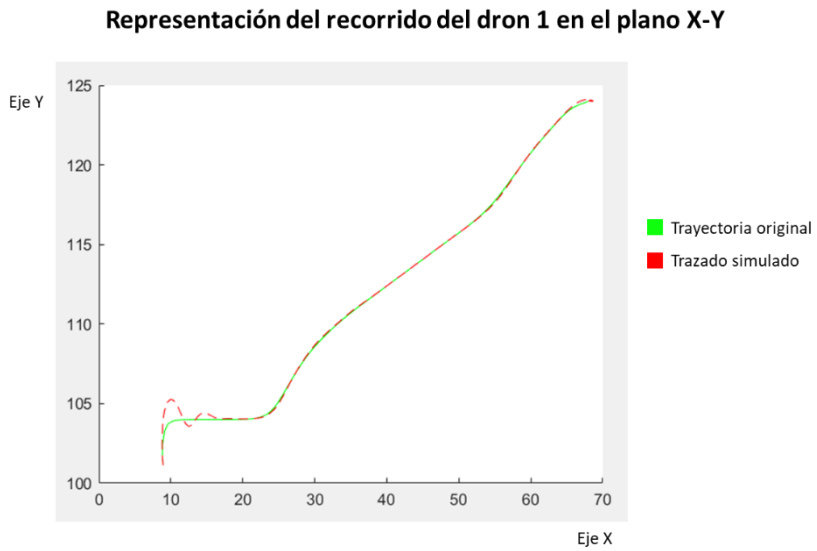

Figura 5.12 Representación del recorrido del dron 1. Misión 1, Caso 3

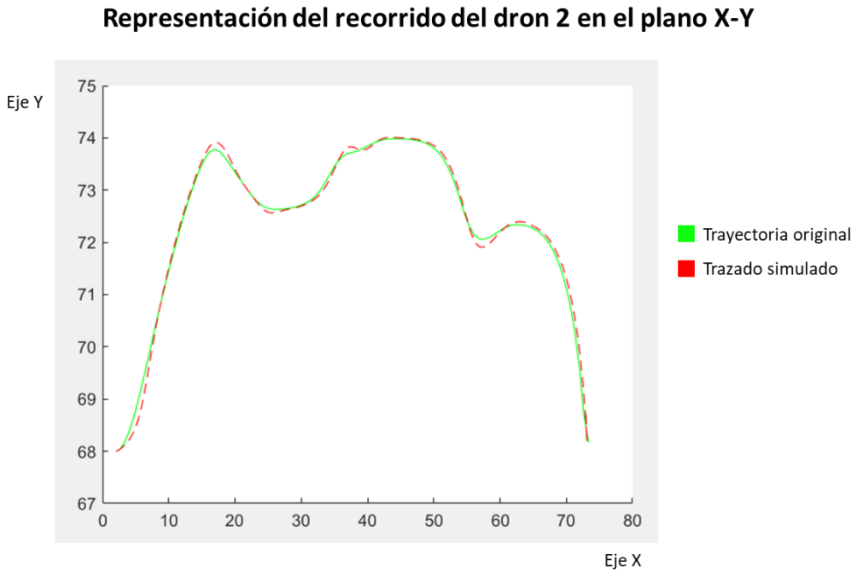

Figura 5.13 Representación del recorrido del dron 2. Misión 1, Caso 3

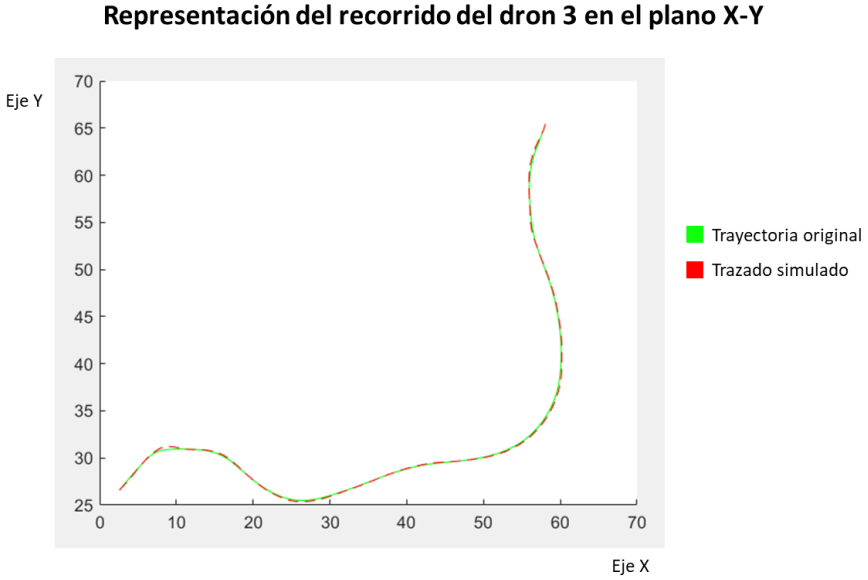

Figura 5.14 Representación del recorrido del dron 3. Misión 1, Caso 3

Con este proceso iterativo de aumentar la velocidad a la par que el radio del PurePursuit, se pueden comprobar los comportamientos del sistema para los distintos valores. Estos cambios, aunque se encuentran en diversos subsistemas del sistema controlador, se han parametrizado, y la modificación de ellos se puede hacer mediante adecuación de los valores de las variables de memoria asociadas a cada elemento que se puede variar.

Tras una serie de iteraciones, como valor final de la velocidad máxima se ha seleccionado $5\,\frac{m}{s}$ que, aunque sin ser una velocidad muy alta, permite que la simulación dure alrededor de 30 segundos, un valor mucho más reducido que el inicial de 170 segundos. Esta disminución del tiempo como se ve es inversamente proporcional con aumento de la velocidad como era de esperar en este caso.

Debido a este aumento de velocidad, los parámetros finales que se van a emplear como referencia serán la velocidad de 5 $\frac{m}{s}$ mencionada, y un radio del *Look ahead distance* de 3.5 metros. El valor del controlador sigue siendo de 1 solo en su parte proporcional. Estas condiciones serán nombradas como el Caso 4.

El resultado de estos nuevos parámetros se puede ver en la siguiente figura. En ella se tiene una representación de tanto la evolución de las coordenadas para cada dron como de los recorridos reales comparados con los planificados en el plano X-Y para los 3 drones. Si se comparan las evoluciones de las coordenadas, se puede apreciar como las curvas descritas tienen una forma casi idéntica a las tomadas como referencia del Caso 1. Esto se puede corroborar con la representación del plano X-Y de cada dron, donde se puede apreciar como las trayectorias se ajustan de una forma relativamente precisa a los valores de la trayectoria planificada.

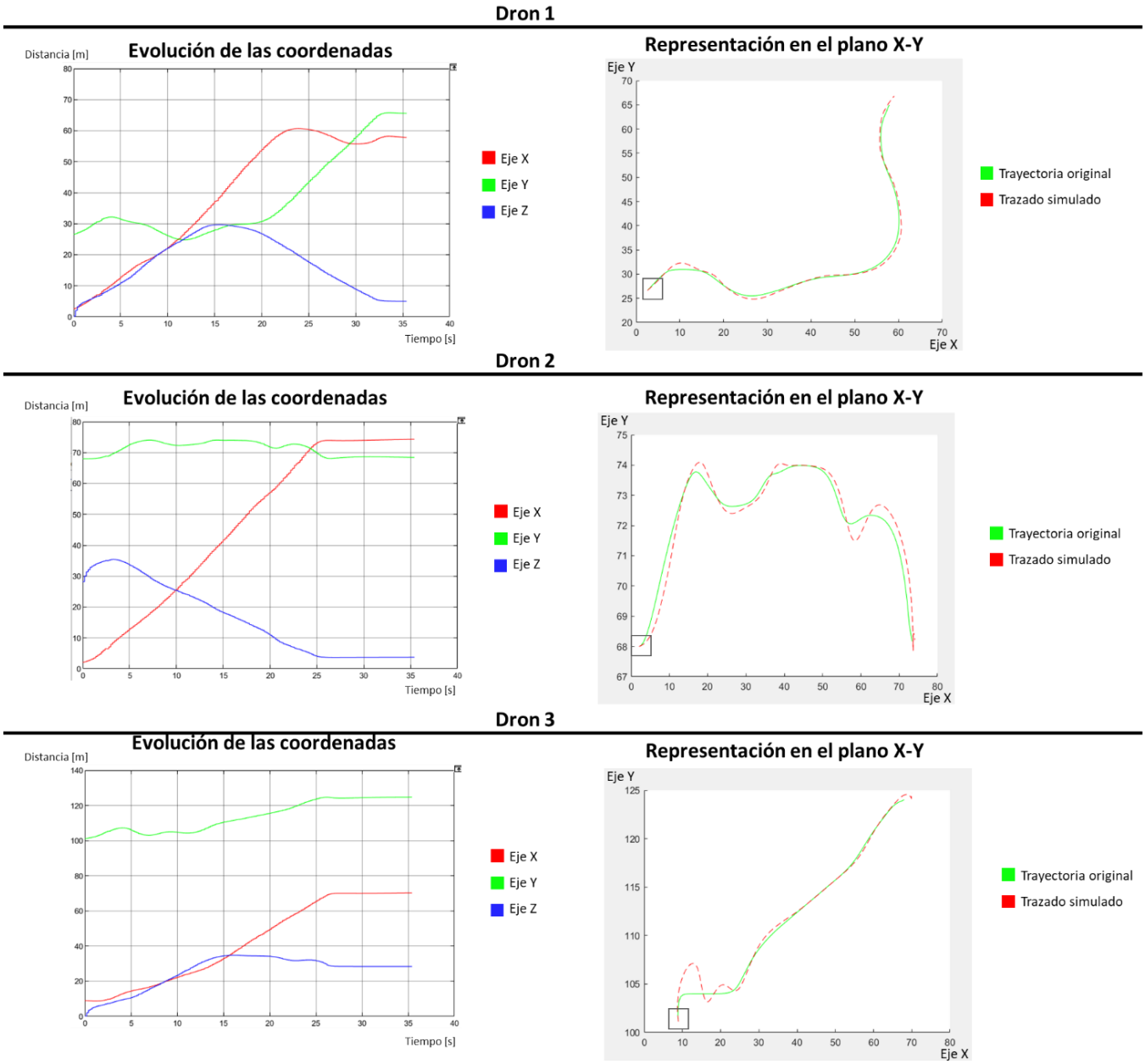

Figura 5.15 Representación de la situación final de la primera misión usando la configuración del Caso 4

Aunque los valores obtenidos son bastante precisos, se puede observar cómo existen puntos en los que la desviación se hace más evidente. Los valores de estas desviaciones se pueden encontrar en la tabla 5.1 junto con las posiciones en las que se dan y los valores esperados y los realmente obtenidos.

Tabla 5.1 Valores de las desviaciones máximas obtenidas en la primera misión

Dron	1	2	3
Coordenada X	10.26	58.49	12.53
Coordenada Y planificada	30.95	72.10	104.10
Coordenada Y simulada	32.31	71.50	107.00
Distancia Máxima	**1.36**	**0.60**	**2.90**

Se puede observar como en el caso de primer y segundo dron los valores de desviación máxima son inferiores al metro y medio de distancia, lo que sumado a que solo se alcanzan estos valores en puntos muy concretos de la trayectoria, se puede considerar como unos valores aceptables. En caso del dron 3 es distinto, ya que cerca del punto inicial del recorrido llega casi a alcanzar los 3 metros de desviación. Este problema se puede considerar menor debido a que ocurre al inicio de la simulación y el seguimiento durante el resto de la trayectoria es incluso mejor que en los otros casos. Además, en la misión 2 se implementará un sistema adicional que resuelve problemas asociados al proceso de despegue en casos problemáticos como este.

En lo que se refiere a velocidades más altas a la finalmente empleada, aunque el sistema tiene la capacidad de aumentarla, esto resultaría un problema en las trayectorias con giros más cerrados al pasar entre edificios. Esta desviación mayor de la trayectoria planificada no solo significaría un mero desvío, si no que podría acabar con la colisión del dron. De hecho, se ha llegado a comprobar que el funcionamiento es aceptable hasta con velocidades de 8 $\frac{m}{s}$ sin que afecte en gran manera a la respuesta del sistema, sin embargo, las pequeñas variaciones que ocurren entre iteraciones de simulaciones generaban, de forma esporádica, situaciones en las que si llegaba a colisionar. Además, al no tener que cumplir unos requerimientos fijos, y con el único fin de demostrar las capacidades del sistema se ha optado por mantener la velocidad en el valor de 5 $\frac{m}{s}$ que otorga una mayor seguridad de que se realiza correctamente y de una forma repetible en el tiempo.

En lo que respecta al controlador, se entiende que para aplicaciones más complejas con velocidades más exigentes sería necesario modificar los parámetros para ajustarse al comportamiento a estas altas velocidades.

Trabajando en este rango de velocidades definitivo para el proyecto, se ha comprobado que el sistema actual, únicamente con parte proporcional es suficiente. Lo que no ha hecho que no se hayan realizado pruebas con otras configuraciones en la búsqueda de un rendimiento mejor.

Al modificar el parámetro D, la parte derivativa, se observaba una reducción notable del rendimiento mostrado. El comportamiento seguía siendo el mismo en el simulador de Gazebo, sin embargo, el tiempo mostrado en Simulink no se correspondía con el tiempo real, por lo que se complica el estudio de los datos a posteriori por no tener la referencia temporal adecuada. Además, esta ralentización del reloj del sistema del controlador no era consistente, por lo que no se podía extrapolar a los tiempos reales con seguridad de mantener los valores exactos.

Además de este problema, los resultados obtenidos no eran mejores que los obtenidos únicamente con la parte proporcional, siendo en ocasiones peores, incluso con valores bajos. Lo mismo ocurría para la parte integral del controlador, que, aunque no ocasionaba los problemas de sincronización, no se obtenían mejoras en el seguimiento de trayectorias que justificaran aplicar estos parámetros.

Como resultado de la simulación de esta primera misión se ha podido determinar los parámetros básicos de velocidad máxima en 5 $\frac{m}{s}$, el radio del PurePursuit en 3.5 metros y el controlador haciendo únicamente uso de la parte proporcional con el valor de 1.

Como visión final de esta misión se muestra la figura 5.16. Esta imagen es una instantánea de la simulación de la primera misión en el entorno de Gazebo. Aunque la representación de los datos en Matlab es mucho más eficiente para analizar datos y comportamientos, se ha visto necesario mostrar la apariencia real del simulador.

En esta figura se puede ver la zona cercana al eje de coordenadas donde se puede ver al dron 1 en un punto intermedio de su recorrido. Además, se ha postprocesado la imagen añadiendo una superposición aproximada del recorrido que realiza este dron durante la simulación con la intención de proporcionar una visión más completa del funcionamiento del simulador en esta memoria.

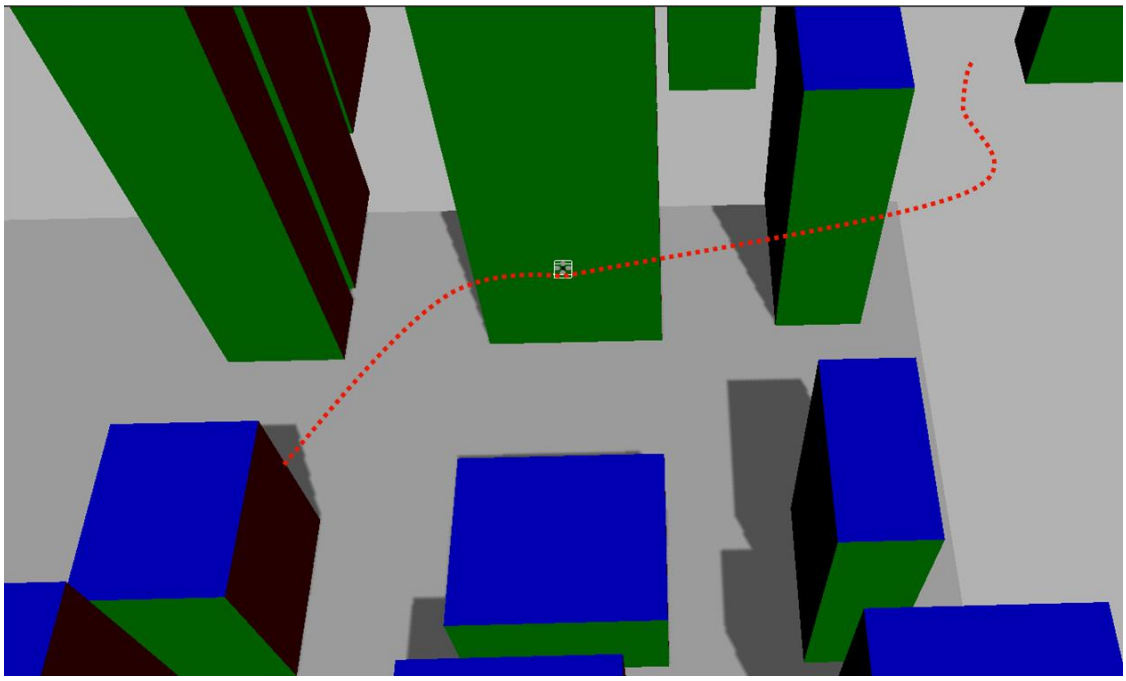

Figura 5.16 Visualización en Gazebo del recorrido del Representación tridimensional de la segunda misión

En las siguientes dos misiones se va a comprobar si estos parámetros se adaptan a nuevos escenarios de una forma satisfactoria y si es necesaria la implementación de modificaciones que mejoren el comportamiento en algún aspecto del sistema.

5.2 Segunda misión. Comprobación de distancias entre drones

En esta segunda misión el objetivo principal es comprobar el comportamiento del sistema en trayectorias con los drones próximos entre sí, haciendo uso de los parámetros finales de funcionamiento determinados en la misión anterior. Para ello, nuevamente se han seleccionado los puntos para tratar de forzar trayectorias que se crucen, para estudiar así su comportamiento.

Las trayectorias que se han seleccionado se pueden observar en las 3 imágenes siguientes ofreciendo los mismos estilos de representación que para el caso anterior. La primera, una vista tridimensional en la que el origen de coordenadas se sitúa en la parte posterior derecha para una visualización más clara. La segunda y tercera, son dos imágenes que se complementan siendo una la vista en el plano X-Y donde se puede situar mejor con relación al escenario anterior y la otra el perfil en el eje Z a lo largo de las trayectorias.

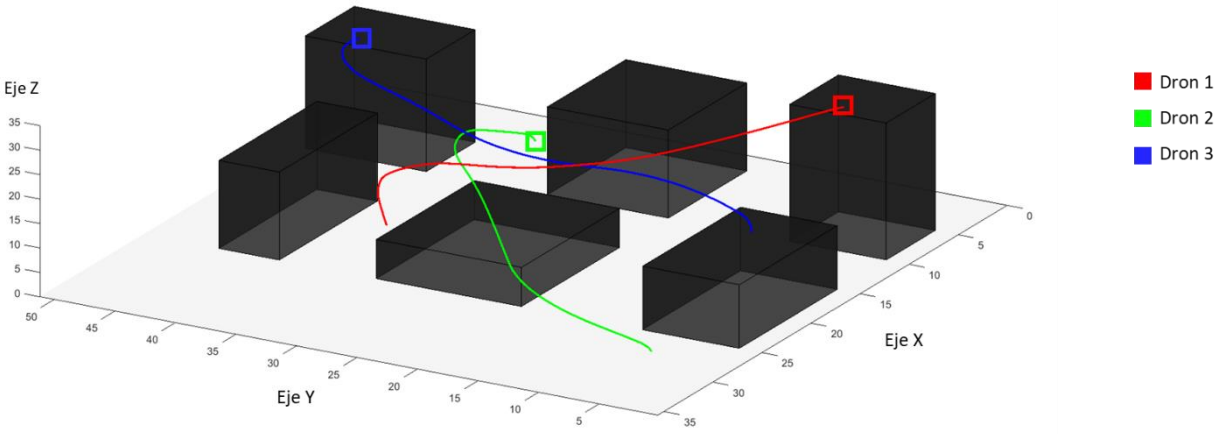

Figura 5.17 Representación tridimensional de la segunda misión

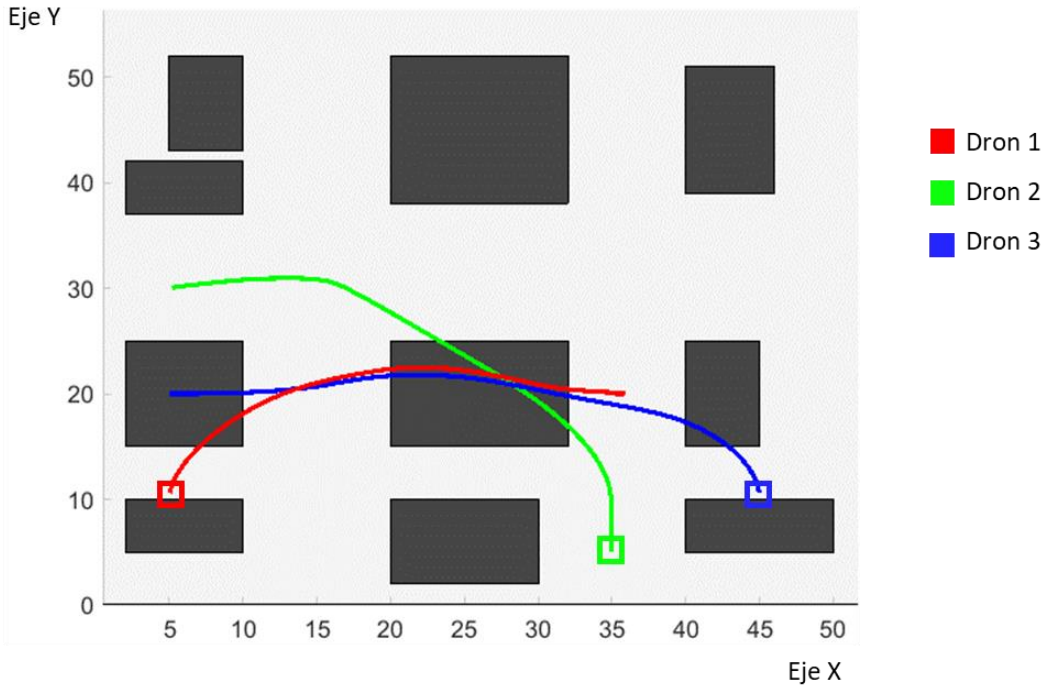

Figura 5.18 Representación en el plano X-Y de la segunda misión

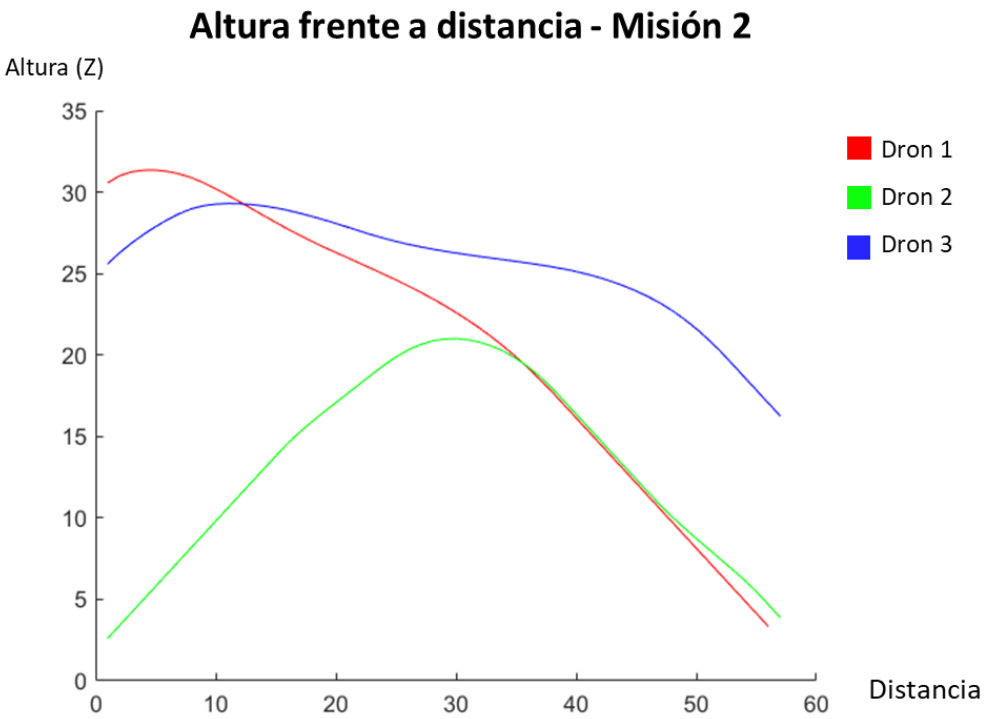

Figura 5.19 Representación de la evolución de la altura en la segunda misión

En estas representaciones se puede observar que en esta simulación no se está haciendo uso de todo el espacio del entorno modelado. Por el contrario, solo se está haciendo uso de la sección de 50x50 metros más cercana al origen de coordenadas. De esta forma, se puede tener en un espacio reducido todos los drones, lo que facilita dos aspectos. El primero es que se hace más sencillo la visualización simultanea de las trayectorias de los drones en el propio simulador. Y la segunda y más importante en esta misión es que de esta forma se pueden forzar más fácilmente la generación de trayectorias próximas entre ellas.

Se puede ver nuevamente que la distribución de los puntos de despegue y aterrizaje se ha distribuido de tal forma que se den todos los casos posibles combinándolos entre el plano del suelo y las azoteas de los edificios.

Hay que notar que, aunque los puntos de despegue de los drones 1 y 3 no se sitúan exactamente encima de los edificios, esto se debe a pequeñas desviaciones en las trayectorias a la hora de ejecutar el algoritmo, pero en la simulación, la posición de inicio de los drones 1 y 3 es realmente en la azotea de los edificios.

Para poder observar correctamente todos los parámetros de esta simulación, se ha hecho necesario añadir funcionalidades nuevas al sistema en Simulink. La principal y más evidente es añadir un sistema que permita determinar las distancias que existen entre los drones.

Aunque esto es algo que se podría postprocesar con todos los datos obtenidos durante la simulación, resulta mucho más sencillo implementar el sistema en el propio controlador, y tiene la ventaja de poder observarlo mientras la simulación se ejecuta. Esto es relativamente sencillo,

dado que basta con unas sencillas operaciones aritméticas con los datos de las posiciones de cada dron para obtenerlas, lo que no supone una carga excesiva al sistema.

Con todos los sistemas anteriores, además del nuevo para la medición de distancias relativas, y haciendo uso de los parámetros del Caso 4, velocidad 5 $\frac{m}{s}$ y radio 3.5 metros se tienen los resultados que se muestran y analizan a continuación.

La primera figura, la 5.20, contiene las tres gráficas referidas a la posición de los drones con su evolución en el tiempo. De ellas se puede extraer el tiempo que tarda cada uno de los drones en realizar sus trayectos, tardando algo menos de 19 segundos el dron 2, con el dron 3 siendo el que menos tarda en completar su ruta, con un tiempo algo inferior a los 16 segundos y el dron 1 empleando un tiempo de 17 segundos. Por otra parte, también se puede leer de estos datos que, en efecto, los valores introducidos al controlador siguen siendo válidos, ya que no se aprecia ninguna señal de oscilaciones repetidas o secciones muy planas en sus evoluciones entre puntos.

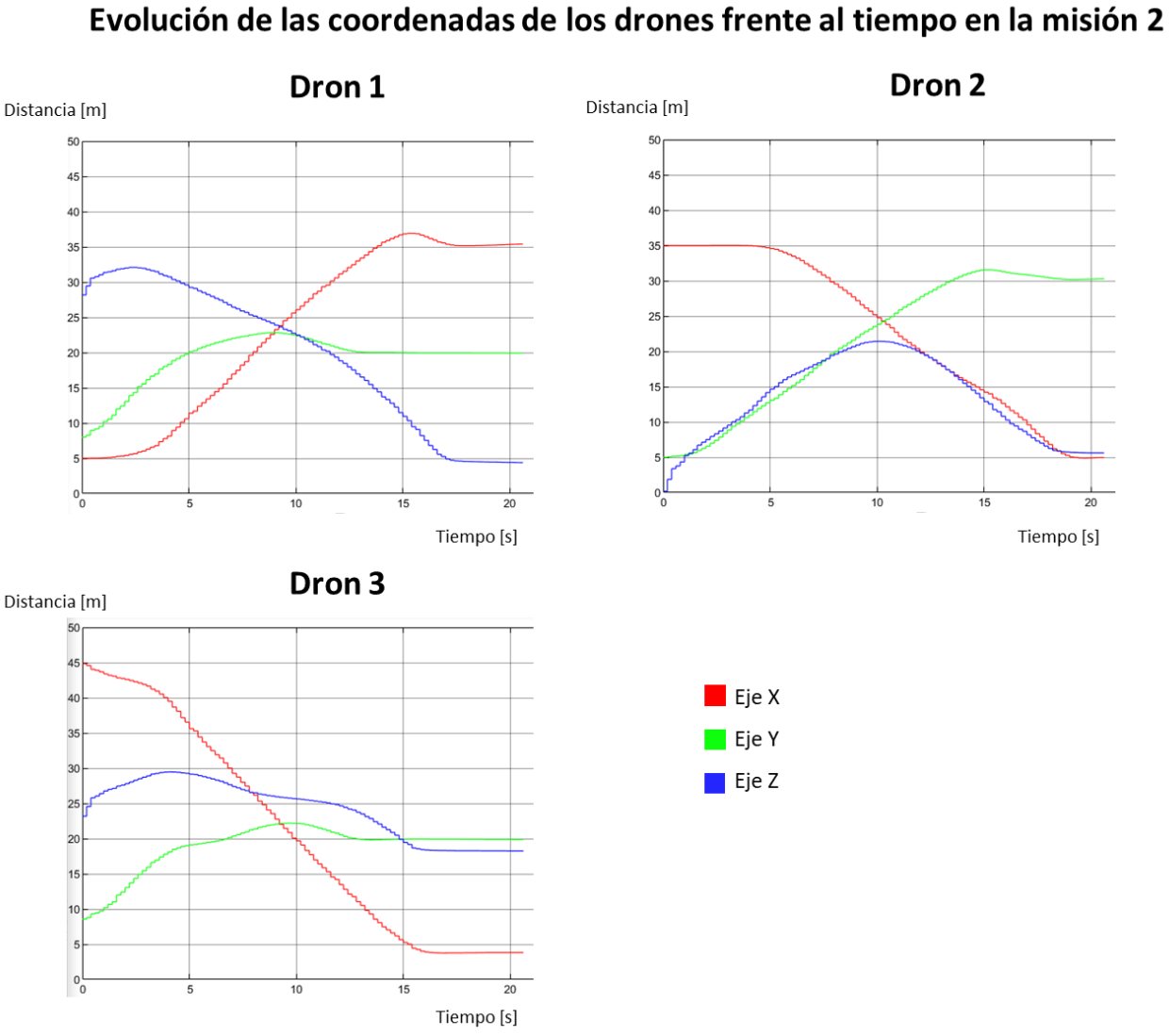

Figura 5.20 Representación de la evolución de las coordenadas de los 3 drones. Misión 2, Caso 4

En la siguiente figura, la 5.21, se tienen representadas las comparativas entre las trayectorias planificadas y realizadas en tres dimensiones para cada dron, en las que nuevamente se ha marcado el punto de inicio con un recuadro para facilitar su entendimiento. En ellas se puede destacar que el dron 1 presenta dos puntos que se desvían notablemente de la trayectoria planificada. El primero cerca del despegue (izquierda de la figura) y el segundo próximo al punto de final de la trayectoria (parte derecha), siendo este último el que mayor desviación presenta, presentado un máximo alrededor de los 1.5 metros.

Para el dron 2 el único punto destacable se produce cerca del punto inicial (en la parte derecha de la figura), aunque esta desviación se puede despreciar por ser menor de medio metro. Esta es una situación similar a la del dron 3, donde su mayor desviación tiene un valor similar, aunque se produce en el giro próximo al final de la trayectoria, en la parte izquierda de la figura correspondiente al dron 3.

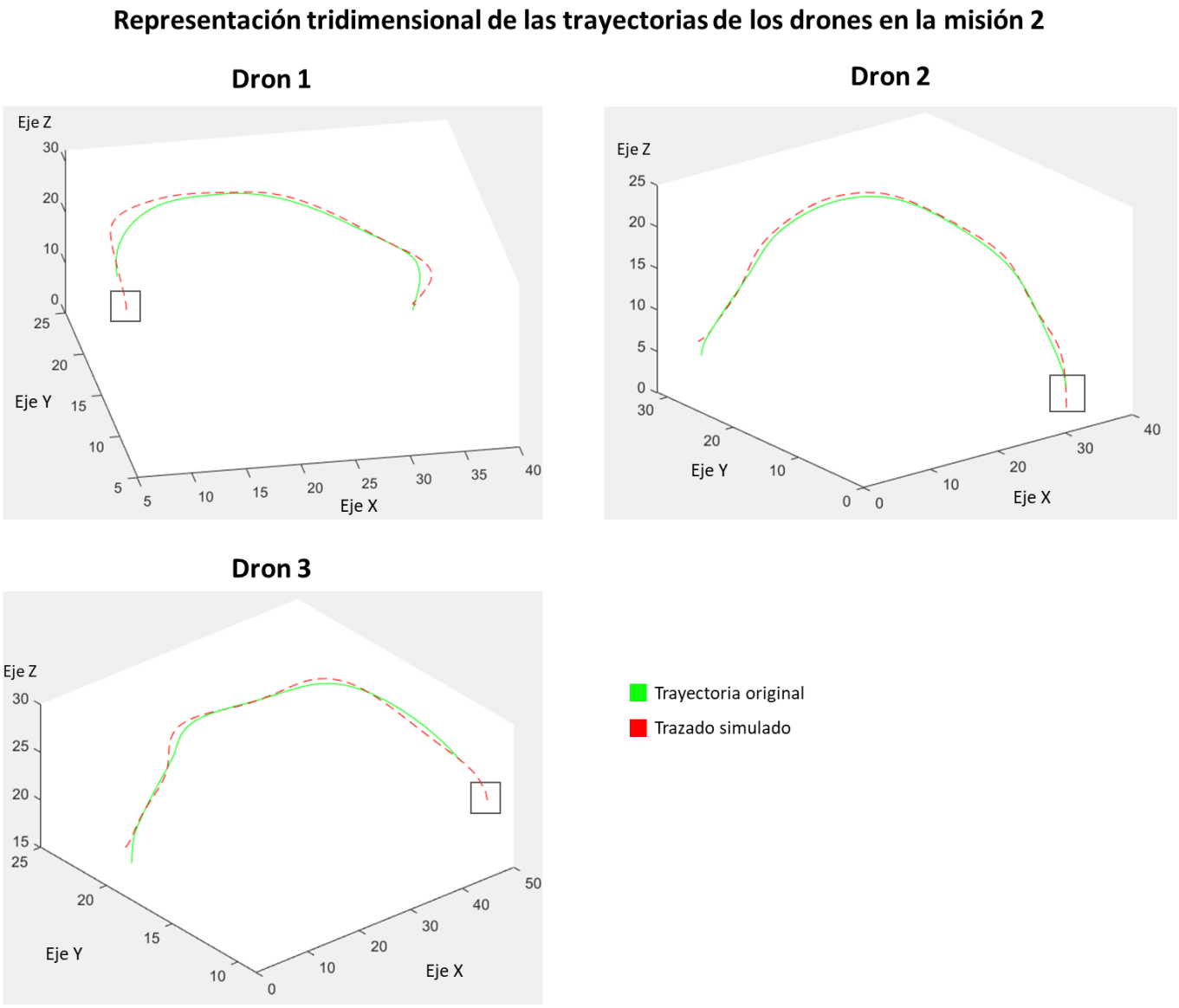

Figura 5.21 Representaciones del recorrido de los 3 drones. Misión2, Caso 4

Con esta segunda visualización se puede comprobar que los parámetros seleccionados durante la primera misión siguen siendo válidos, ya que los comportamientos de los drones son buenos a lo largo de la mayor parte de las tres trayectorias.

La figura 5.22 mostrada a continuación tiene un especial interés, ya que hasta el momento no había sido relevante obtener el grafico de distancias relativas frente al tiempo. En este caso, aunque no es inmediatamente evidente, se ha vuelto a representar haciendo uso de los tres colores RGB, siendo estos los representativos de las distancias entre las parejas de los drones 1 y 2, 1 y 3, y por último el 2 y 3 en este orden para los tres colores representativos. Además, en esta representación se ha representado el valor mínimo en color negro para poder identificarlo de forma más sencilla, así como el cambio de la pareja con menor valor. De esta forma, la manera de identificar que pareja de drones se trata, es mirar cuál de los 3 colores no está presente en el instante analizado, lo cual es sencillo por solo tratar con 3 parejas de datos.

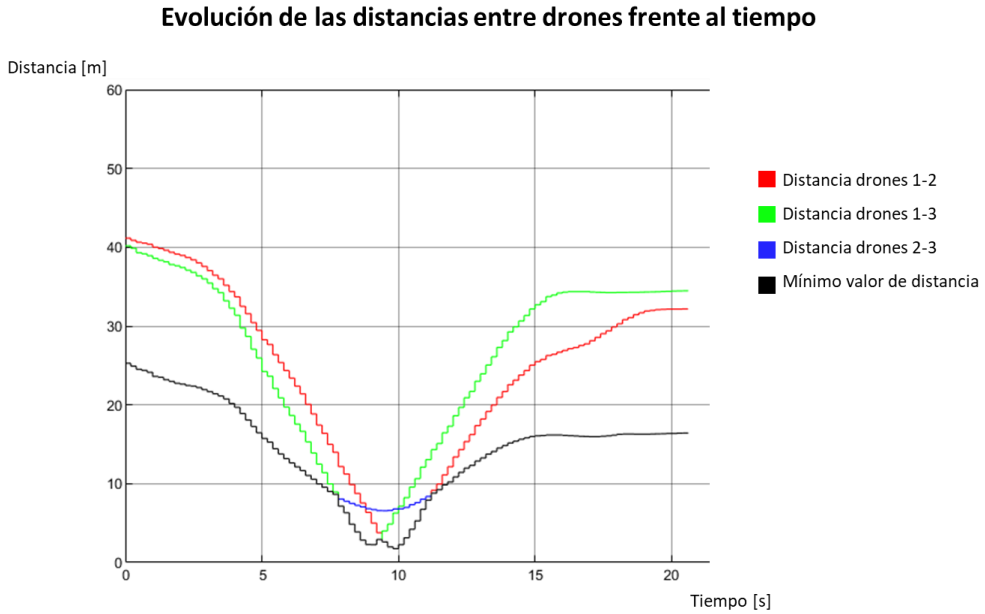

Figura 5.22 Detalle de la evolución de las distancias relativas con el sistema inicial. Misión 2, Caso 4

En esta figura se puede ver cómo es inicialmente a la pareja azul, de los drones 2 y 3, la que más próxima se encuentra, mismo resultado que en la parte final de la simulación. Esto era de esperar, que sus posiciones de inicio y final son ambas las más cercanas entre sí. Por otro lado, en la parte central se alternan entre la pareja verde, de los drones 1 y 3, al principio, y la roja, de 1 y 2 después como las más cercanas. Son estas zonas las que entrañan un mayor interés ya que son los puntos de mayor proximidad entre drones en toda la simulación.

El detalle de esta zona se puede ver en la figura 5.23. En ella se observa cómo se mediante el empleo de los cursores de las gráficas en Matlab se puede determinar con precisión el instante y el valor de estos dos puntos críticos. Los valores quedan recogidos en su tabla inferior, la 5.2.

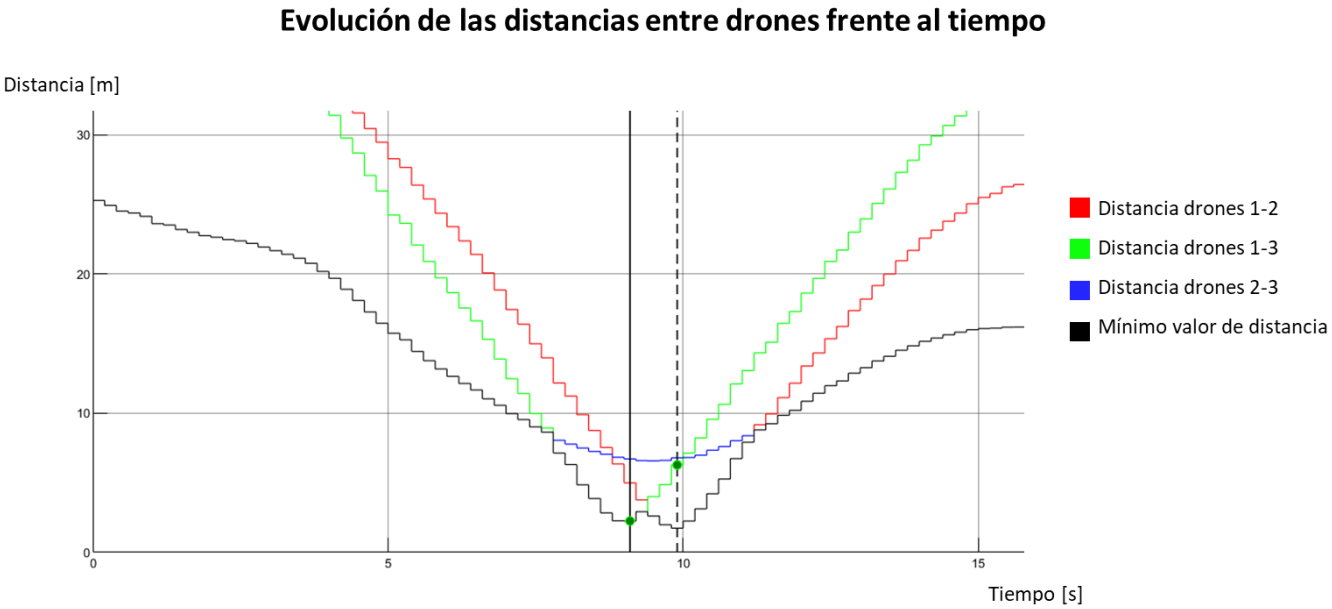

Figura 5.23 Detalle de la evolución de las distancias relativas. Misión 2, configuración inicial

Tabla 5.2 Valores de distancias mínimas. Misión 2, Caso 4

Tiempo	Distancia	Pareja
9.1 segundos	2.256 metros	Verde (1-3)
9.9 segundos	1.747 metros	Roja (1-2)

Se puede observar cómo es la primera pareja la que recoge la distancia mínima de toda la simulación con un valor de algo menos de 1.80 metros, seguido de los 2.25 de la pareja verde instantes antes.

Estos valores no representan un problema, especialmente al comprobar que la máxima desviación de los movimientos realizados respecto a las trayectorias calculadas no llega a superar el metro de distancia. Sin embargo, en una situación real podría llegar a resultar un problema si se añadiera alguna perturbación externa. Es por este motivo que se ha decidido implementar una modificación al controlador para minimizar este riesgo.

De esta forma se pretende obtener un resultado más seguro, a la vez que se demuestra la flexibilidad del sistema a la hora de incluir modificaciones. Para ellos se han implementado dos sistemas adicionales distintos.

El primero es una mejora del proceso de despegue de los drones, haciendo que si la distancia al primer punto es superior a radio de búsqueda del PurePursuit se suavice, además de posibilitar realizarlo a una velocidad menor. De esta forma se ha implementado un sistema inicial que acerca al dron hasta el primer punto con el objetivo de suavizar los primeros instantes de simulación para aquellos casos en los que la distancia sea significativa como ocurría con el dron 1 en la primera misión.

En el caso de esta simulación nuevamente solo afecta, y de forma muy reducida, al recorrido del primer dron, ya que el comportamiento del resto de drones no tiene especiales problemas para realizar este primer tramo del recorrido. No obstante, el efecto de esta implementación será más notable durante la simulación de la tercera y última misión que se desarrolla en el sub apartado siguiente donde se podrá ver una mejora sustancial.

El otro punto que se ha mejorado es la opción de reducir la velocidad de los drones en los instantes en los que la distancia entre ellos disminuya por debajo de un cierto valor. Para el caso de esta misión, el valor de la distancia se ha seleccionado en 10 metros, distancia a partir de la cual se reduce de forma gradual la velocidad máxima permitida de las aeronaves implicadas. La velocidad será menor a medida que su distancia también lo es, volviendo a la normalidad cuando se alejan más allá de la distancia seleccionada.

Por simplicidad se ha hecho un sistema que varía la velocidad de forma proporcional, pero si fuera necesario podría cambiarse para que esta disminución tuviera otro comportamiento más complejo, y tuviera además en cuenta otros parámetros como el signo de la variación o su ratio de cambio. En este proyecto con el único propósito de mostrar las capacidades del simulador se ha mantenido con una implementación sencilla.

Tras volver a realizar la simulación de la misma misión con el sistema controlador nuevo y los parámetros anteriores se han obtenido unos trazados en el recorrido aproximadamente iguales, debido a que ya se ajustaban bastante bien al original, por lo que no se mostraran nuevamente.

Donde sí se tiene una mayor diferencia es en el perfil de distancias relativas. Esta nueva prueba mantiene los mismos parámetros del sistema anterior, pero incorpora el nuevo controlador, por lo que será referida como Caso 5. Los resultados del perfil de distancias se pueden ver en la figura 5.24 junto con la tabla 5.3 donde se muestran los valores numéricos de los mínimos relativo y absolutos.

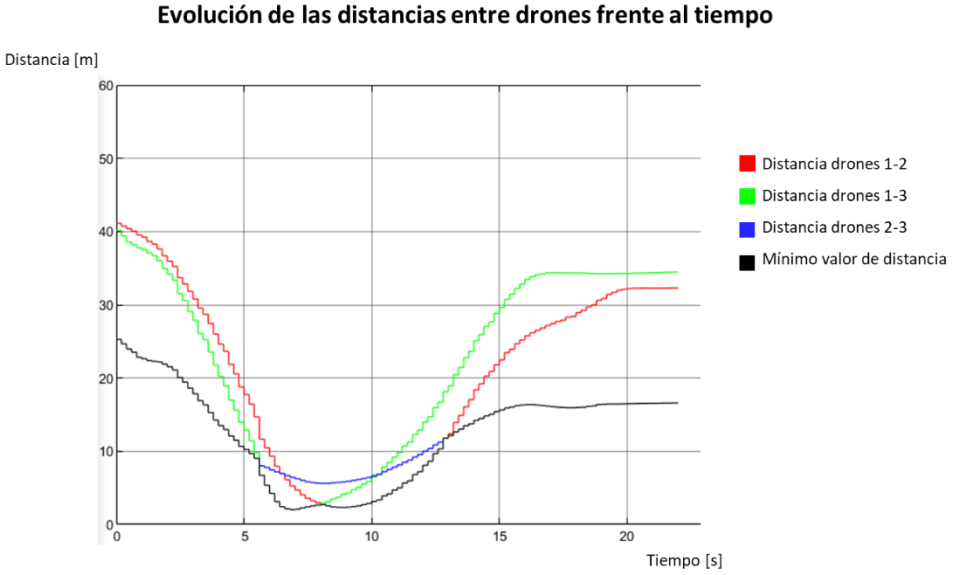

Figura 5.24 Evolución de las distancias relativas con el sistema mejorado. Misión 2, Caso 5

Tabla 5.3 Valores de distancias mínimas. Misión 2, Caso 5

Tiempo	Distancia	Pareja
6.9 segundos	2.039 metros	Verde (1-3)
8.9 segundos	2.326 metros	Roja (1-2)

En la figura se vuelve a tener una estructura muy similar a la del caso anterior, en la se vuelven a tener dos mínimos en el punto central de la simulación que mantienen el orden en el que se dan. Sin embargo, hay dos aspectos importantes que sí que varían.

El primero es que la simulación es ligeramente más larga, ascendiendo a un tiempo de 20.5 segundos, algo esperado por reducir las velocidades de los drones. Por otra parte, se puede observar como la parte critica del trayecto se expande en el tiempo, abarcando en esta segunda prueba alrededor de los 6 segundos siendo algo superior al doble que para el caso anterior.

En cuanto a los valores numéricos de estas distancias, para la pareja verde se ha disminuido la distancia mínima en 0.217 metros, mientras que para la pareja roja se ha aumentado la distancia mínima en 0.579 metros.

El valor de la segunda pareja se ha aumentado considerablemente siendo un tercio mayor que en el primer caso, mientras que para la otra pareja el valor se ha visto reducido en un 10%. Aunque menor, sigue estando por encima de los dos metros en todo momento y solo baja a esa distancia en un periodo de tiempo muy pequeño en comparación con el total de la simulación.

Además, otro factor que se tiene que considerar, es que estos acercamientos se realizan a una velocidad menor, lo que facilita el seguimiento de la trayectoria, disminuyendo los posibles errores de trazado. Esto se pueden ver en la figura 5.25, donde se recoge la evolución de las velocidades con el tiempo.

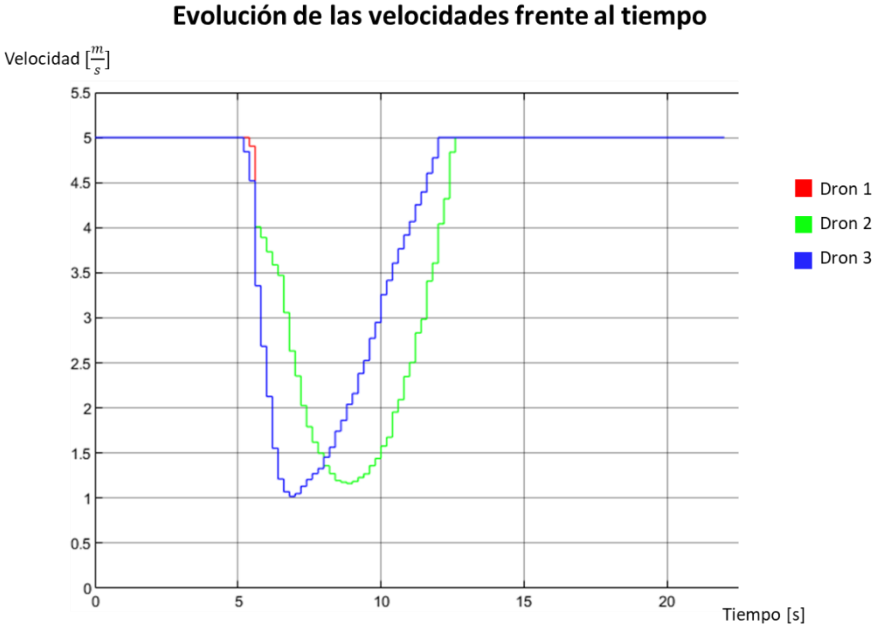

Figura 5.25 Evolución de las velocidades frente al tiempo. Misión 2, Caso 5

En la imagen se puede ver como las velocidades tienden a estar en el máximo de 5 $\frac{m}{s}$ establecido, pero tienen bajadas hasta casi 1 $\frac{m}{s}$. Estos picos se puede comprobar que se corresponden con los valles en las gráficas de distancias relativas. El trazado del dron 1, se ve afectado por ambos acercamientos, con lo que su valor se corresponde siempre con el mínimo valor en ambos valles. Se ha preferido representar superpuesto el valor de los otros dos drones para poder diferenciar mejor el dron adicional que se ve involucrado en cada momento.

En esta misión se aprecian los efectos de una mejora que, aunque pequeña es notable y puede hacer que la estabilidad del sistema sea mayor en aquellos puntos donde se pudiera ver comprometida. Con las modificaciones se ha pasado de hacer un control fijo y constante a uno dinámico que automáticamente varía sus propios parámetros de funcionamiento para ajustarse de la mejor forma posible al plan de vuelo preestablecido en cada momento.

Nuevamente, en la figura 5.26 se tiene una instantánea de Gazebo en mitad de la simulación de esta segunda misión. Se corresponde con el primer instante crítico, donde se produce el acercamiento máximo entre los drones 1 y 2. Nuevamente, se han superpuesto las trayectorias aproximadas de los drones para tener una idea general de toda la simulación.

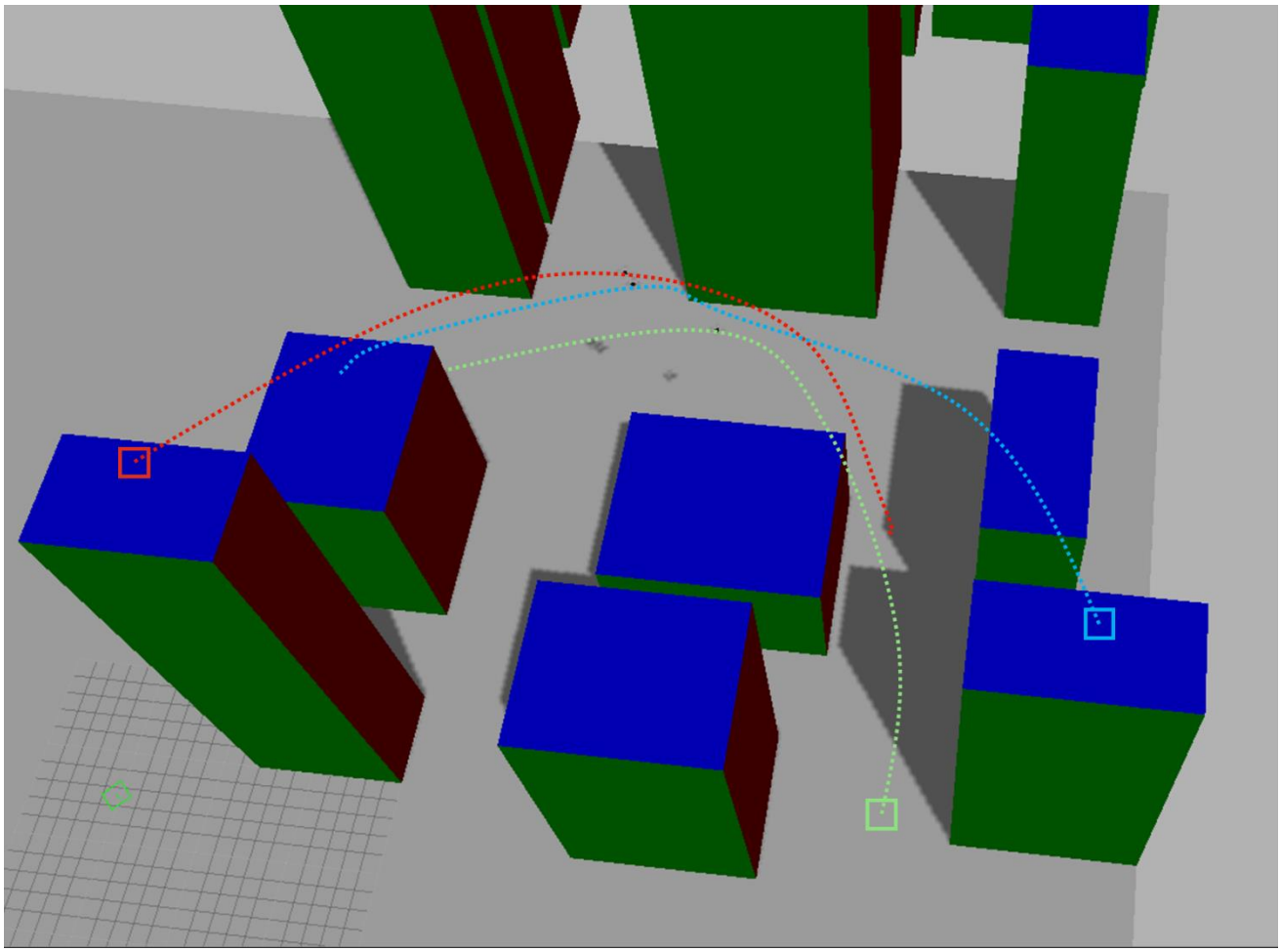

Figura 5.26 Visualización en Gazebo del primer instante critico de la misión 2

5.3 Tercera misión. Interacciones en vuelo entre drones

Para este tercer y último escenario de simulación se pretende volver a forzar al sistema para obtener trayectorias cercanas y poder observar el comportamiento de esta forma. En este caso nuevamente se emplearán 3 drones con trayectorias centradas en el área reducida de 50 x 50 metros empleada en la simulación anterior.

En este caso, se ha llevado más al extremo aun, ya que las trayectorias de los drones 1 y 2 tienen los mismos puntos de inicio y final pero invertidos, mientras que el tercero se pretende simplemente hacer pasar por un punto próximo como en el caso anterior. El resultado del cálculo de estas trayectorias en este caso es peculiar, ya que su procesamiento de forma independiente tiene como salida un resultado distinto del que se obtiene si se realizan de forma simultánea.

En la figura 5.27 se puede ver el resultado de su cálculo de cada una de forma individual y posteriormente superpuesto en una misma representación. Para obtenerla se ha ejecutado tres veces el algoritmo con uno solo dron en cada caso, usando las coordenadas correspondientes. Como se puede observar, las trayectorias de los drones 1 y 2 son prácticamente idénticas en su recorrido tridimensional. Lo cual es un resultado esperable por realizarse entre los mismos puntos del espacio usando el mismo algoritmo de cálculo. Sin embargo, cuando se procesan las tres trayectorias simultáneamente de la forma habitual, el resultado es distinto.

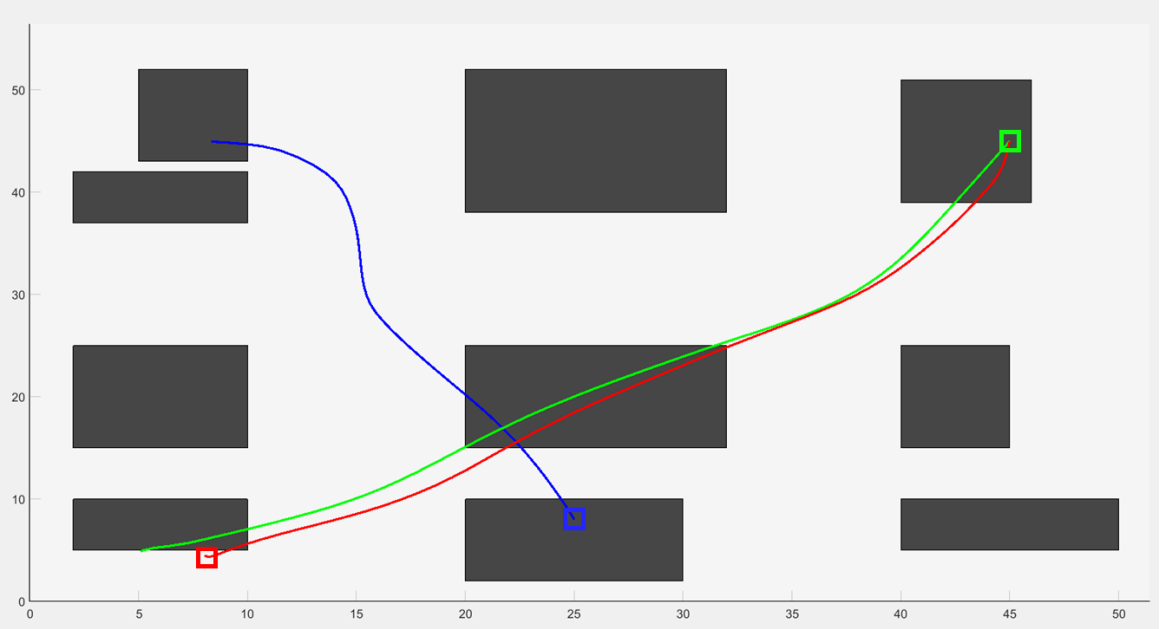

Figura 5.27 Representación en el plano X-Y de la tercera misión de las trayectorias individuales

Este otro resultado completo se muestra en la figura 5.28. En ella se puede apreciar como la trayectoria del primer dron, la línea roja, se desvía del trazado original. Esto es debido a que como se comentó en el apartado *4.1 Obtención de las trayectorias*, se tiene en cuenta a los drones como obstáculos a la hora de calcular las trayectorias. En este caso, y de forma arbitraria, el dron dos se ha marcado con una prioridad mayor que la del primer dron, por lo que es el dron 1 el que altera su trayectoria para no impedir el paso. En la figura 5.29 se puede ver esta misma escena representada en tres dimensiones, donde el origen de coordenadas se encuentra en el frontal de la imagen.

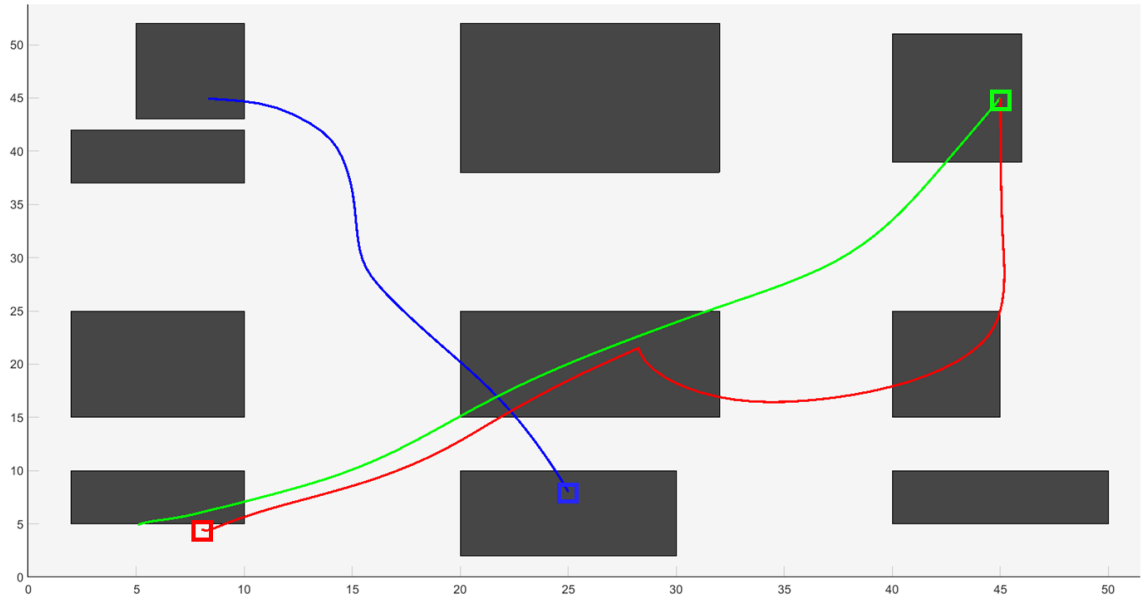

Figura 5.28 Representación en el plano X-Y de la tercera misión

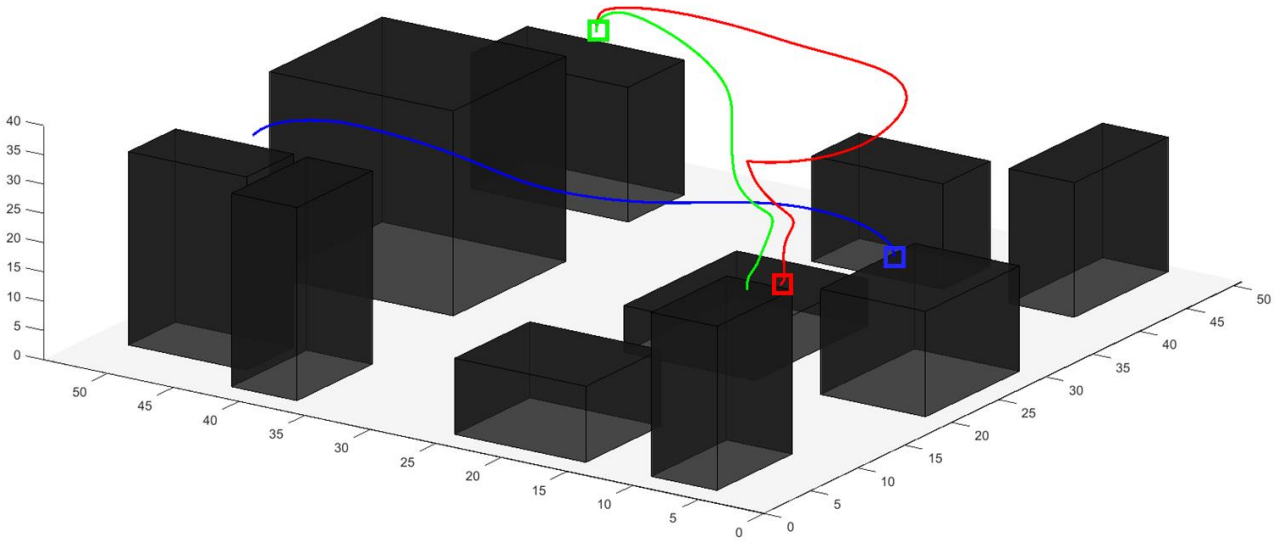

Figura 5.29 Representación tridimensional de la tercera misión

Haciendo uso de este último cálculo de las trayectorias se va a proceder a mostrar los resultados obtenidos durante la simulación. En primer lugar, se emplea el sistema de control sin ningún tipo de modificación, el Caso 4, como en el caso de la primera misión para después compararlo con el sistema obtenido hacia el final de la segunda misión, incluyendo todos los sistemas adaptativos de velocidad, el Caso 5.

Los resultados obtenidos de la simulación con el sistema de control original se muestran a continuación en la figura 5.30. En las representaciones en tiempo real de la evolución de las coordenadas se puede observar como el comportamiento de los drones 2 y es bastante uniforme. No obstante, se aprecia como en la parte inicial del recorrido del dron 1 tiene oscilaciones en las tres coordenadas, hasta que en el segundo 10 se estabiliza y continua el resto de la simulación de forma más estable.

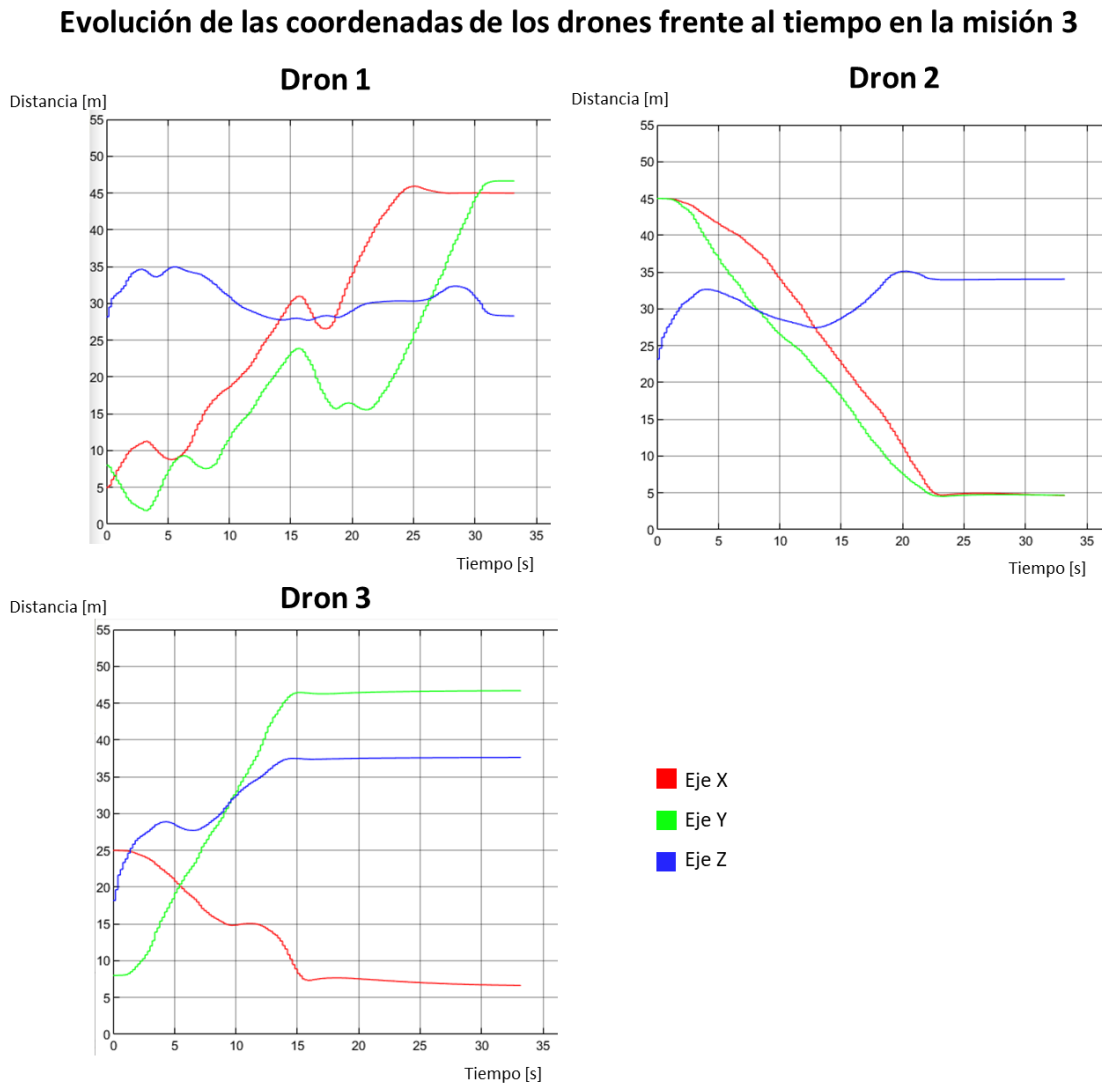

Figura 5.30 Representación de la evolución de las coordenadas de los 3 drones. Misión 3, Caso 4

Esto se puede comprobar al revisar las representaciones tridimensionales de las trayectorias representadas a continuación en la figura 5.31. En ellas se puede apreciar como los drones 2 y 3 se ajustan a las trayectorias de forma bastante precisa, mientras que el primero tiene grandes desviaciones en las que trata de ajustarse de forma errática a la trayectoria de referencia marcada en verde. Una vez la alcanza se mantiene al nivel de precisión observado en los otros drones, pero el inicio es bastante impreciso.

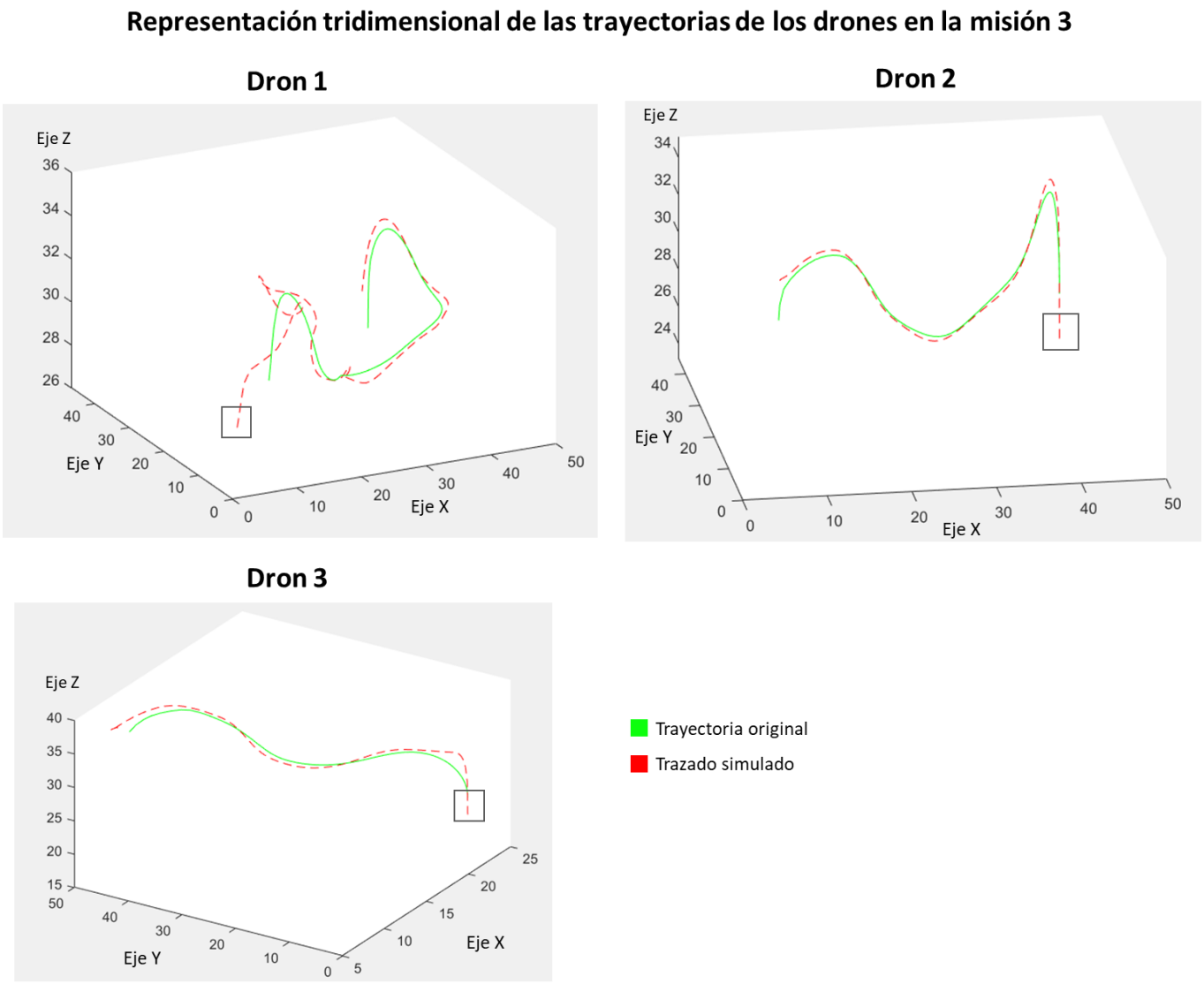

Figura 5.31 Representación tridimensional de las trayectorias de los 3 drones. Misión 3, Caso 4

Este error inicial se puede asumir que es derivado de una distancia relativamente que existe entre la posición inicial del dron y la primera coordenada de la trayectoria. Además, existe un gran cambio de dirección que se debe realizar instantes después de iniciar el movimiento. Esto, como se verá al analizar el sistema mejorado obtenido en la misión anterior se resolverá en gran medida gracias a las modificaciones en el proceso de despegue de los drones.

En lo que se refiere a las distancias entre drones, se pueden ver los resultados obtenidos tras esta primera prueba en la figura 5.32. En ella se puede apreciar como las distancias en general son bastante elevadas entre las parejas de drones, sin bajar de los 10 metros. Sin embargo, cerca de los 14 segundos se produce una bajada que llega a situar a la pareja verde, de los drones 1 y 3, a una distancia críticamente baja, de tan solo 1.114 metros.

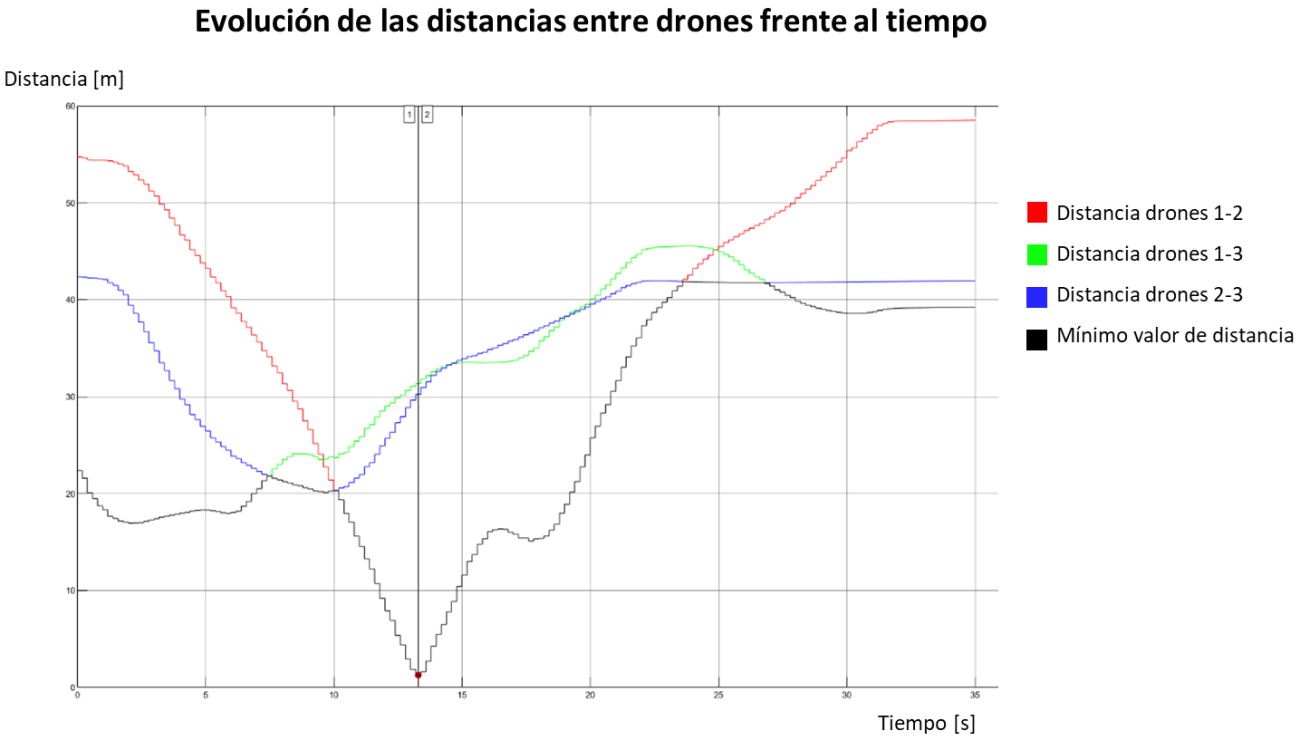

Figura 5.32 Detalle de la evolución de las distancias relativas. Misión 3, Caso 4

Tabla 5.4 Valores de distancias mínimas. Misión 3, Caso 4

Tiempo	Distancia	Pareja
13.7 segundos	1.114 metros	Verde (1-3)

Este es un valor bastante menor a los obtenidos en la misión anterior, por lo que, desde el punto de vista de la seguridad de la ejecución no sería totalmente adecuado. No obstante, si se analiza la simulación cuidadosamente, se debe a que el comportamiento de los drones no es el esperado. Donde al principio se podía intuir que la trayectoria del dron 1 se apartaba para dejar paso al dron 2 se entiende que este segundo dron debería estar en la zona anterior al punto donde las trayectorias divergen. Sin embargo, en esta simulación no ocurre así, ya que ambos coinciden en el punto donde el dron se desvía.

Es por este motivo, por lo que se hace necesario regular la velocidad de forma dinámica para evitar llegar a alcanzar distancias de separación tan bajas y forzar el comportamiento adecuado para el dron.

Tras incorporar el sistema avanzado de despegue se puede ver como el comportamiento inicial del dron 1 mejora mucho respecto a mostrado en la figura 5.31 anterior. En la nueva figura 5.32 se puede ver como se ajusta de una manera mucho más precisa a la trayectoria calculada. El sistema, además de facilitar un punto intermedio, debido a la gran distancia, ha reducido su velocidad a la mitad para facilitar el desarrollo de esta fase inicial de la simulación que resultaba muy conflictiva para el primer dron.

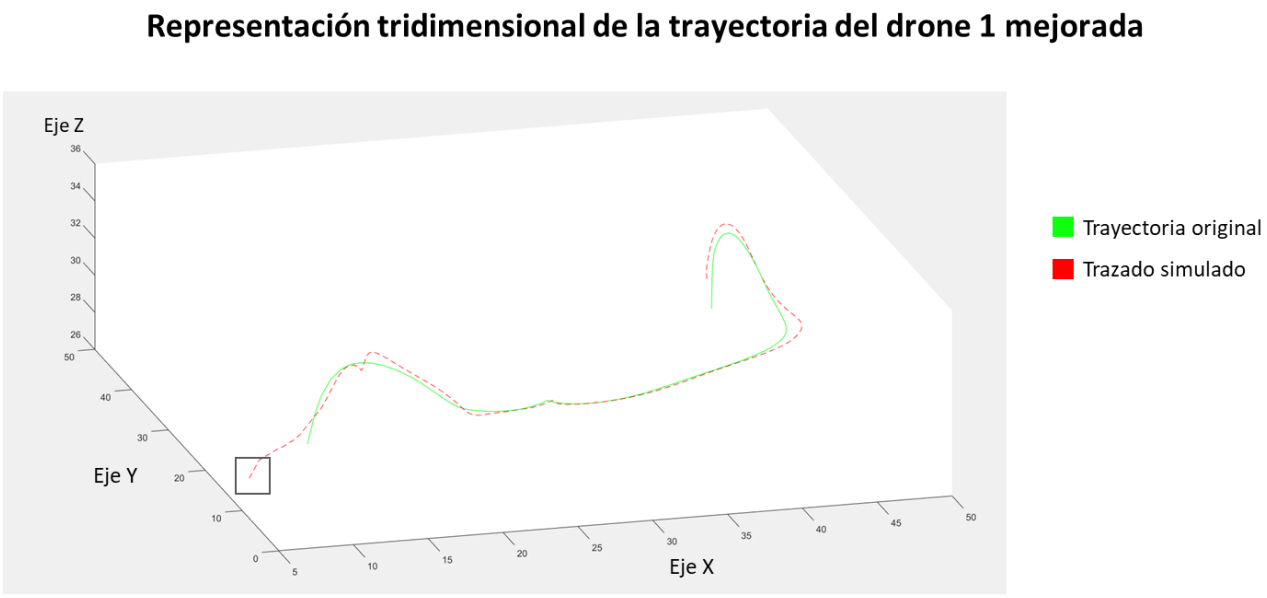

Figura 5.33 Representación tridimensional de la trayectoria del dron 1. Misión 3, Caso 5

Esta modificación, también se puede apreciar en la figura 5.34 donde se muestran los perfiles de velocidad de esta última simulación. En los primeros instantes, inferiores a 1 segundo, se puede ver como la velocidad en rojo del primer dron comienza en 2.5 $\frac{m}{s}$ para luego llegar hasta los 5 definida como su velocidad máxima.

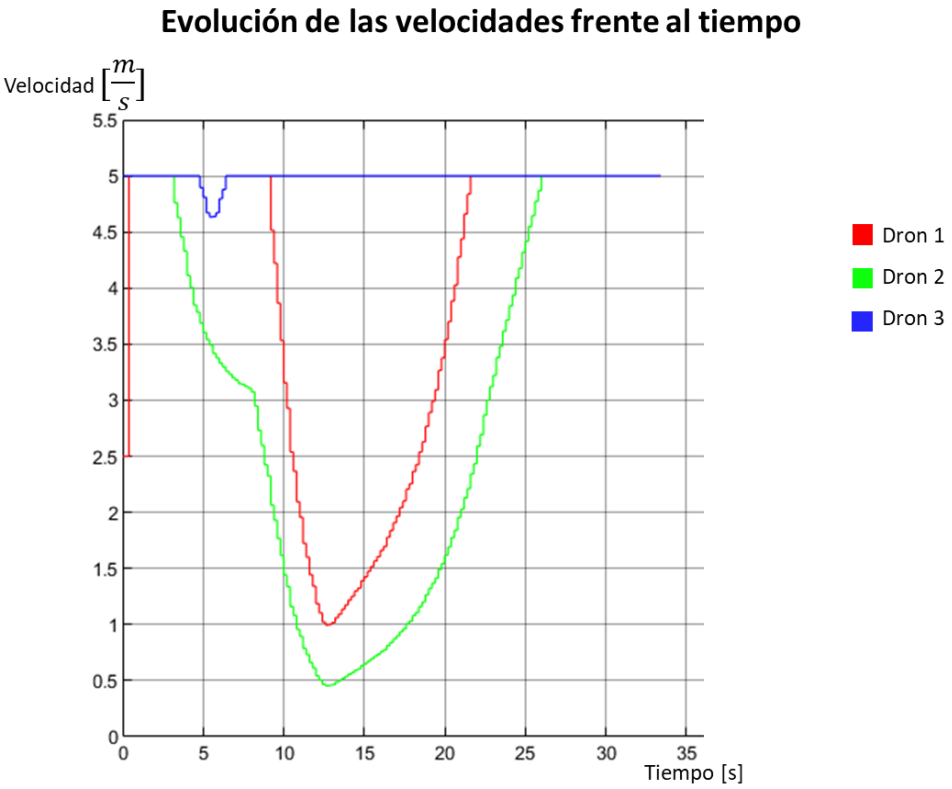

Figura 5.34 Evolución de las velocidades. Misión 3, Caso 5

En esta figura también se puede apreciar como durante el desarrollo de la simulación ocurren dos picos en los que las velocidades disminuyen. El primero afecta a los drones 1 y 3. Este se debe al acercamiento inicial entre estos drones. En la figura 5.35 donde se muestran las distancias relativas se puede apreciar como en esos primeros instantes la pareja de drones 1 y 3 es la que tiene la mínima distancia de separación, pero debido a su valor relativamente alto, su velocidad apenas se ve reducida. Tras esto, la velocidad máxima de los drones 1 y 3 vuelve rápidamente a su valor original.

Por su parte, la del dron 2 comienza a reducirse debido a que continúa aproximándose al primer dron. Este segundo dron comienza a reducir la velocidad a pesar de que se encuentra a una distancia grande, esto es debido a que el valor de distancia umbral se ha establecido en un valor mayor.

En este caso es así porque se pretende de esta forma adecuar el comportamiento, y conseguir que no se acerque al punto de desvió del dron 1 hasta que este se haya comenzado a retirar. Se puede observar como la velocidad del dron 1 baja hasta los 1.1 $\frac{m}{s}$, mientras que la del dron 2 se ve reducida hasta los 0.5 $\frac{m}{s}$, asimetría debida a esta diferencia en la ponderación de las distancias umbrales.

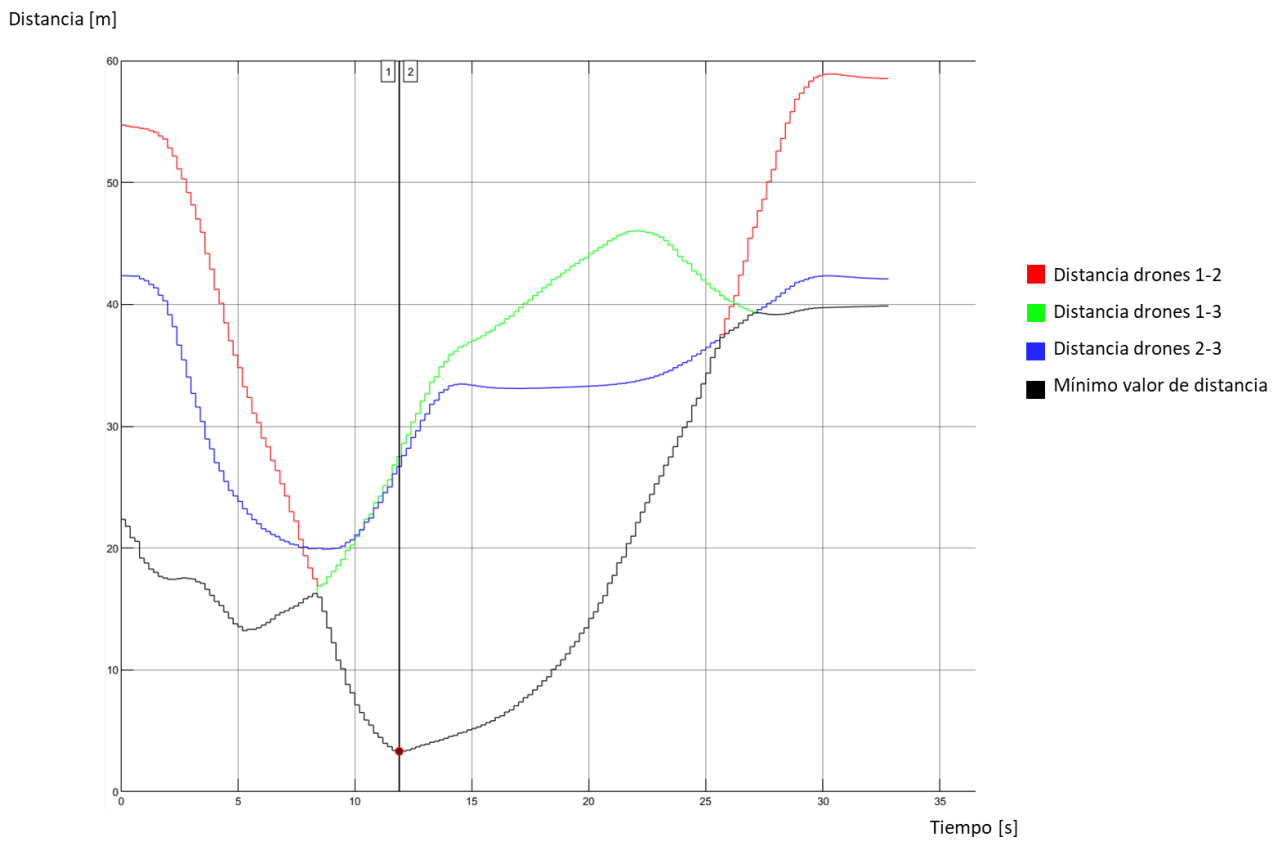

Figura 5.35 Detalle de la evolución de las distancias relativas. Misión 3, Caso 5

Tabla 5.5 Valores de distancias mínimas. Misión 3, Caso 5

Tiempo	Distancia	Pareja
11.9 segundos	3.324 metros	Roja (1-2)

Con esta configuración, el mínimo absoluto de distancias se sitúa en los 12 segundos con un valor de 3.3 metros como se muestra en la figura 5.35. Un valor mucho más favorable desde el lado de la seguridad no solo por el hecho de ser un valor casi tres veces mayor, sino porque además, éste se da con velocidades muy inferiores a las iniciales, por lo que la maniobra de aproximación se realiza con muchas más garantías de éxito.

Por último, se muestra la figura 5.36 que se corresponde a la visualización en Gazebo de la simulación de esta última misión. Se puede ver el instante previo al que el dron 1 modifica su trayectoria para apartar y dejar paso al dron 2. Se trata de una simulación con la versión final del controlador, lo que se puede apreciar por la mayor distancia que existe entre ambos drones. Nuevamente las trayectorias se han superpuesto a la imagen del simulador para poder entender fácilmente la situación, lo que es especialmente útil con el dron 3 que se encuentra fuera de plano y más alejado del foco de la imagen.

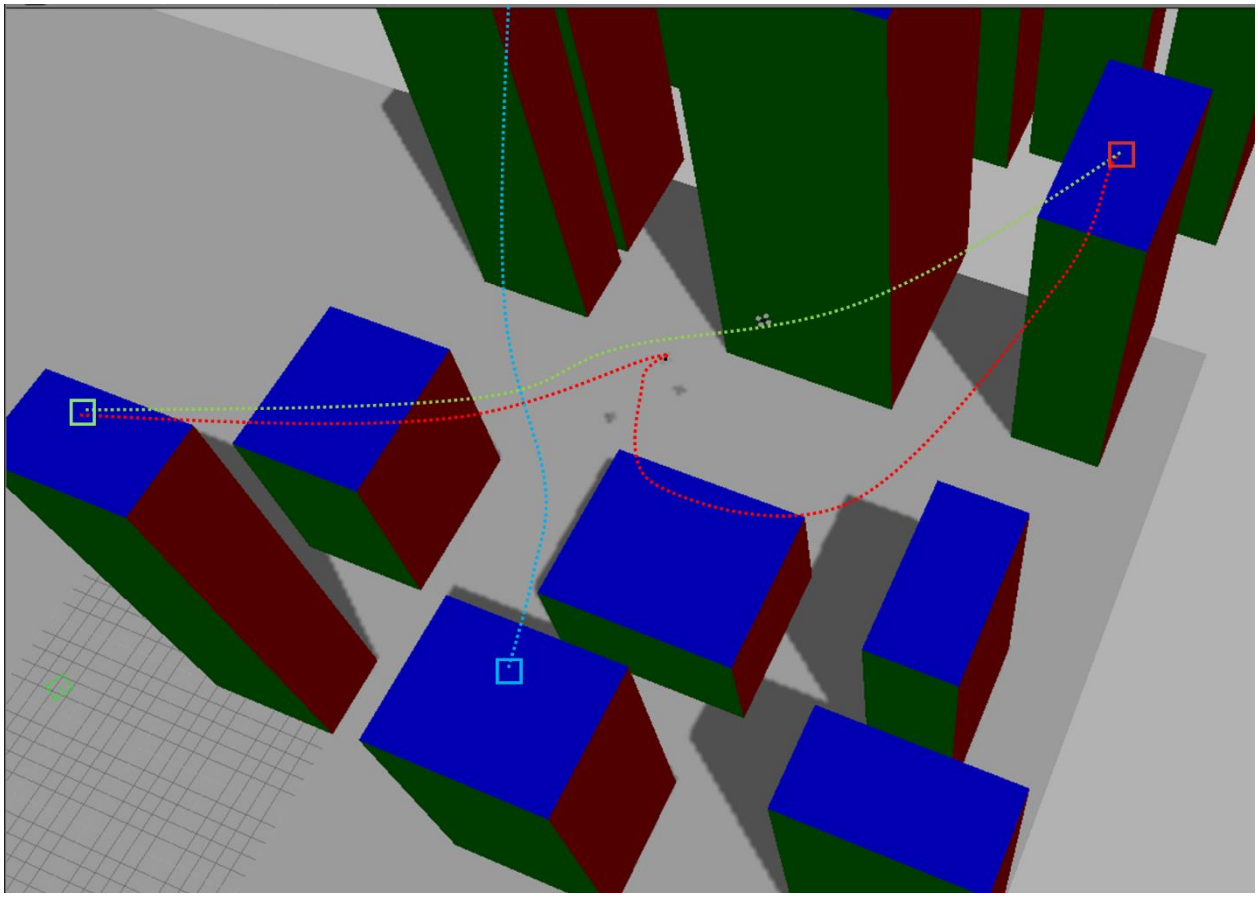

Figura 5.36 Visualización en Gazebo del primer instante critico de la misión 3

Con este análisis de las misiones simuladas se da por concluido el capítulo de resultados. Debido a que el proyecto se basa en la construcción de un sistema para programar y simular vuelos con quadcopters, la forma que se ha considerado más adecuada para mostrarlo es mediante la simulación y visualización de varias misiones con distintas condiciones.

Durante la el proceso de simulación, se ha podido comprobar el funcionamiento y comportamiento de los drones de forma experimental mediante la visualización proporcionada por el propio simulador. Sin embargo, la observación de imágenes esporádicas del proceso simulado no proporciona mucha información por sí misma, por lo que, con el fin de poder analizar y presentar los datos con precisión, se han incluido los sistemas de recogida de datos que han hecho posible todas las representaciones de datos que se han mostrado.

Estas figuras con las representaciones de la evolución de los distintos parámetros permiten no solo estudiar los distintos valores puntuales, sino que además posibilitan tener una idea general de la forma en la que se desarrollan las distintas misiones en el simulador.

Para completar esta memoria los últimos capítulos se centran en recapitular toda la información relativa al desarrollo del proyecto, su planificación y sus implicaciones socioeconómicas, además de otros en los que se recopilan las conclusiones obtenidas durante todo el proceso y futuras modificaciones y mejoras.

CAPÍTULO 6: Marco socioeconómico

En este sexto capítulo del proyecto se va a tratar dos aspectos importantes como son la planificación y las fases que definen el proyecto, así como el presupuesto estimado que se le puede computar a todo el proceso de elaboración de este.

6.1 Planificación

El inicio de este proyecto tiene fecha del mes de octubre de 2019, cuando se comenzó a definir lo que terminaría siendo este proyecto, y se ha extendido hasta el mes de mayo del año 2020 cuando se ha dado por concluido.

- La primera tarea necesaria para el desarrollo de este proyecto fue la realización de un estudio previo del estado del arte de los algoritmos de planificación de trayectorias para drones, en particular los centrados en el sistema de Fast Marching.
- La segunda tarea consistió en el estudio y comprensión de la aplicación del algoritmo empleado. Esta comprensión previa fue necesaria para realizar su modificación con el fin de poder obtener los datos necesarios para el proceso de simulación, así como poder implementar un sistema de previsualización propio dentro de Matlab.
- La tercera tarea tuvo como principal objetivo tener una instancia del sistema operativo Linux con versiones operativas de ROS y Gazebo. En esta tarea se incluyen otros aspectos asociados como el estudio de la documentación de Gazebo, la creación de los modelos 3D del entorno empleado, además de realizar las modificaciones al entorno Gazebo para poder simular drones de forma adecuada.
- La cuarta tarea se centra en el entorno de Simulink, ya que el principal objetivo de esta fase fue la creación de un sistema capaz de coordinar los datos de todos los sistemas implicados. En esta tarea se incluye además la creación del sistema de control que, haciendo uso de toda la información disponible, transmite los comandos necesarios al simulador para que los drones realicen los movimientos requeridos.
- La quinta tarea del proyecto se centró en el estudio del simulador final. Mediante numerosas simulaciones se han podido estudiar diversos escenarios, comprobando y refinando el comportamiento y funcionalidades del sistema. Durante esta fase del proyecto se han modificado aspectos que atañen a los diversos elementos que componen el simulador, entre las que se incluyen desde la fase de planificación de trayectoria hasta el sistema de visualización final.
- La sexta y última tarea es la referida al desarrollo de esta memoria, en la que se incluye no solo la redacción del documento, sino la extracción de los datos e imágenes más relevantes y explicativos que se han mostrado durante todo el documento.

6.2 Presupuesto

En este apartado del presupuesto se pretende hacer un resumen de todos los medios que han sido necesarios para realizar el proyecto, tanto medios físicos incluyendo el hardware y software empelado, así como las horas de trabajo empleadas.

En la tabla inferior se muestran los elementos referidos al Hardware y software necesarios para el proyecto con sus precios. Se ha estimado la vida útil del portátil y de su correspondiente licencia de Windows en 5 años para hacer el cálculo del gasto asociado al proyecto. En el resto de licencias se incluye por un lado las gratuitas, entre las que hay dos tipos, las que se trata de versiones con fines no comerciales, y las que son software abierto. Por otro lado, la licencia de Matlab tiene un carácter de suscripción, por lo que junto con el resto del software se ha estimado su periodo útil en un año debido a las frecuentes actualizaciones con nuevas características.

Tabla 6.1: Coste de hardware y software

Descripción	Coste	Dedicación (meses)	Vida útil (meses)	Coste atribuible (Euros)
Software				
Windows 10 Home	145	8	60	19.33
Matlab and Simulink Student	69	8	12	46
Blender	0	8	12	0
VMware non commercial licence	0	8	12	0
ROS	0	8	12	0
Gazebo	0	8	12	0
Kate	0	8	12	0
			Total Software	65.33
Hardware				
Lenovo Legion Y720	1400	8	60	186.66
			Total Software	168.66
			TOTAL	252

El último aspecto importante del presupuesto es el coste atribuible a las horas dedicadas al desarrollo del proyecto y la presente memoria. Esto se recoge en la tabla inferior.

Tabla 6.2: Coste Horas trabajadas

Descripción	Horas dedicadas	Coste por hora (€)	Coste total (€)
Jefe de Proyecto	35	35	1225
Ingeniero Junior	350	20	7000
		Total	8225

Si se suman ambos valores, queda una estimación de precios costes totales de 8477 € para todo el proyecto.

CAPÍTULO 7: Conclusiones

En este capítulo de la memoria se van a recoger las conclusiones extraídas a modo de resumen durante el desarrollo del proyecto, destacando aquellos aspectos alcanzados y que se han podido lograr implementar.

En general, se puede decir que los objetivos principales planteados se han alcanzado de una forma satisfactoria. El primero de ellos era la planificación de trayectorias haciendo uso del método Fast Marching. Para ello se ha hecho uso de una implementación de este algoritmo modificándolo con la finalidad de poder previsualizar de la forma más adecuada posible las trayectorias, y por ende, los comportamientos que se previstos para realizar por los drones quadcopters.

Esta primera previsualización se considera que ha sido todo un éxito, ya que permite una visualización clara tanto de las trayectorias obtenidas, como del entorno en el que se emplazan. Estas visualizaciones no son solo figuras estáticas, sino que son representaciones tridimensionales que se pueden rotar ampliar, y en definitiva observar con detalle. Esto se ha realizado con el programa Matlab sin tener una carga apreciable sobre la ejecución original del algoritmo, por lo que los tiempos adicionales que se requieren para visualizar la escena final con todos sus detalles son virtualmente nulos. Esto es de un gran valor ya que permite la edición de las condiciones iniciales de la simulación de una forma rápida y eficaz.

Por otra parte, se ha realizado la representación del entorno urbano, principalmente constituido por altos edificios, en el entorno del simulador elegido. Gazebo ha sido el Software en el que se ha llevado a cabo esta simulación, y pese a sus limitaciones, ha permitido la representación del entorno de una forma semiautomatizada, además de incorporar el objeto principal de estudio, los propios drones.

Mediante la implementación de modelos tridimensionales creados para representar los edificios, se ha podido obtener una representación personalizada del entorno. Esta se puede variar en tamaño y composición de los edificios mediante scripts en Matlab que permite modificar el entorno de una forma sencilla con solo modificar las condiciones iniciales del problema planteado.

La inclusión y simulación de los propios drones en Gazebo, aunque no ha sido un proceso trivial, se ha podido realizar gracias a las adaptaciones creadas por ROS. Estas han resultado clave para realizar simulaciones de estos vehículos aéreos en el entorno del simulador.

El aspecto fundamental en el proceso de creación del simulador es la interacción entre los sistemas. Este es otro punto que puede parecer trivial y se puede pasar por alto por no ser una parte tangible del proyecto. Sin embargo, ha resultado uno de los más problemáticos.

La falta de documentación referente a algunos problemas que surgen cuando se trata de comunicar los distintos programas es algo que resulta difícil de paliar, y en numerosas ocasiones se tiene que recurrir como único recurso a investigar el propio código original de los sistemas. Por suerte, una vez establecida esta conexión y comprobar que los datos pueden viajar en ambos sentidos, es una parte del proyecto que no es necesaria modificar para realizar modificaciones a los escenarios simulados.

Esta dificultad por suerte contrasta con la fluidez de uso de Simulink para la creación del controlador de vuelo. En él se utiliza una programación por medio de sencillos bloques que permite una fácil comprensión incluso para aquellos no tan familiarizados con la programación de código tradicional. Esta facilidad de uso permite la adicción y modificación de parámetros de una forma relativamente sencilla y rápida, que permite crear prototipos para el proyecto rápidamente.

Estas agrupaciones de bloques sencillos se han combinado en estructuras que se pueden reutilizar. Mediante la incorporación de variables como parámetros se pueden modificar sin variar la propia estructura de la aplicación. Con esto, se consigue alternar entre las distintas funcionalidades implementadas como son la asistencia al despegue o la regulación de velocidad con la proximidad.

Y es tras implementar toda la parte operativa del proyecto cuando se puede realmente obtener los resultados a los que se enfocaba el proyecto desde un principio. El conjunto es el resultado de una estructura entrelazada que trabaja entre distintas aplicaciones para dar como resultado una simulación ordenada y coherente de la situación elegida.

Otro aspecto muy importante es el tiempo de previo de inicialización de la simulación final. Este se encuentra alrededor de los 30 segundos, en los que se incluye el reinicio de los elementos del simulador Gazebo, y la compilación y puesta a punto de los parámetros del controlador en Simulink. Si se realiza de forma simultánea, el tiempo es más reducido que si se suman por separado, ya que el proceso de reinicio del simulador, aunque no es instantáneo no requiere de gran carga para el procesador. El cuello de botella de éste es la compilación y ejecución del sistema de control. En definitiva, el tiempo de carga no es un aspecto que tenga demasiado peso, en especial si se simulan misiones más largas ya que este factor no influye en la carga.

En el entorno del simulador, la información que se puede obtener es cualitativa, se puede observar el funcionamiento y comportamientos del sistema. Para la obtención de los datos concretos de lo que está ocurriendo se tiene que hacer uso de Matlab y analizarlos mediante el graficado de los mismos. Estas series temporales de datos se pueden mostrar y analizar como cualquier otro conjunto de números con todas las herramientas que Matlab ofrece como se ha mostrado con las figuras de los capítulos anteriores.

Estas representaciones pueden llegar a ser muy detalladas y contener información muy útil, en especial las representaciones tridimensionales, por el carácter de este proyecto. Estas representaciones sin embargo a la hora de mostrarlas en un formato bidimensional pierden gran parte de la profundidad en la información que presentan. Sin embargo, en su estado original pueden llegar a ser mucho más intuitivas que las representaciones de dos ejes para este tipo de tareas.

En cuanto a los valores obtenidos, son en general buenos por la forma en la que se ajusta los datos obtenidos a los planificados originalmente. El aspecto más comprometido puede ser la distancia entre los propios drones, pero esto ocurre generalmente en un instante de tiempo pequeño frente al total de la simulación.

Además, gracias los sistemas dinámicos de control de la velocidad en función de la distancia entre drones, se ha logrado aumentar los valores, o al menos mantenerlos mientras se reducía de forma momentánea la velocidad. De esta forma se consigue tener unas condiciones de mayor seguridad ante las posibles colisiones entre drones.

Otro objetivo de las simulaciones es la repetibilidad. Con la configuración final obtenida del simulador se tiene un sistema que reproduce los movimientos con una fidelidad suficiente para amortiguar las pequeñas variaciones que ocurren de forma normal en las simulaciones. Como en la realidad, en el simulador también existen pequeñas variaciones en el comportamiento de los sistemas entre simulaciones de las mismas condiciones iniciales. Gracias a la obtención de un sistema bastante estable, no existen variaciones significativas entre repeticiones como si ocurre cuando el comportamiento de los drones es más errático y no se ajusta debidamente a la planificación original.

A modo de resumen, se puede decir que se ha conseguido un sistema funcional con todas las necesidades básicas de un simulador para quadcopters que ha demostrado tener la capacidad de simular, observar y analizar distintos entornos y condiciones adaptables para abordar un amplio abanico de escenarios, de una forma satisfactoria y reproducible.

CAPÍTULO 8: Trabajos futuros

En el apartado de trabajos futuros y ampliaciones para este proyecto, se van a proponer aquellos aspectos en los que se podrían añadir funcionalidades o mejorar las existentes.

La primera ampliación está relacionada con la determinación de las trayectorias. Adaptar la implementación de Fast Marching para obtener rutas actualizadas a medida que la simulación avanza sería una compleja, aunque muy interesante ampliación. El algoritmo empleado tiene en cuenta el desarrollo que tienen el conjunto de los drones, pero debido a las desviaciones que se pueden dar respecto al plan establecido podría ser interesante realizar nuevos cálculos a medida que la simulación avanza. Además, se podría modificar para actualizar también el entorno, modificándolo y añadiendo en tiempo real obstáculos que se puedan detectar mediante las posibles lecturas de sensores incorporados en los drones. Además, de esta forma se podría ajustar los instantes iniciales de la trayectoria de una forma mucho más precisa.

El segundo punto que se podría mejorar es el relacionado con la visualización dentro del entorno de Gazebo. Debido a que los drones son tan pequeños en comparación con el escenario completo, se pueden llegar a perder de vista si se tiene una visión muy general y alejada del entorno.

Debido a problemas con la versión empleada del simulador, el escalado de los modelos complejos como es el del dron, no está correctamente implementado, por lo que no se ha podido mostrar con un tamaño mayor del real. Otra opción alternativa seria tratar de hacer que el dron fuera visible a través de los edificios viéndose destacado, aunque se encontrara tras ellos, como se trató de implementar en la previsualización de Matlab.

La tercera mejora que se propone es la representación en el entorno de Gazebo de la trayectoria seguida. En la versión empleada esto no es posible y se requiere de otras aplicaciones como Rviz para lograrlo. Poder representar estos datos en el propio entorno de simulación resultaría en una gran mejora en la capacidad de análisis en tiempo real, especialmente si también se pudiera representar otras trayectorias como la planificada originalmente.

La última mejora que se propone es la de encontrar un sistema que permita el control de la cámara del simulador de forma programada. En la versión empleada de Gazebo se tiene la opción de seguir a cualquiera de los modelos de la simulación, sin embargo, este seguimiento no se realiza de forma adecuada en el caso de los drones. Esto probablemente se deba a que no está diseñado para elementos que se desplacen en las tres dimensiones del espacio grandes distancias, por lo que el sistema termina encuadrando al modelo sin mantener un ángulo fijo enfocándolo generalmente desde su parte inferior lo que hace difícil la visualización adecuada y se tiene que recurrir a un control manual.

Estos son los aspectos que se consideran más importantes para las futuras ampliaciones del proyecto. Sin duda el campo de la simulación, y en especial de la de los vehículos aéreos es un campo que tiene una importancia creciente y se seguirán realizando más proyectos en la materia.

CAPÍTULO 9: Bibliografía

[1] S.M. La Valle, “Combinatorial Motion Planning” en Planning Algorithms, Cambridge University Press. 1ª Edición. Cambridge University Press, 2006. [En línea]. Disponible en: http://lavalle.pl/planning/

[2] S.M. La Valle, “Sampling-Based Motion Planning” en Planning Algorithms, Cambridge University Press. 1ª Edición. Cambridge University Press, 2006. [En línea]. Disponible en: http://lavalle.pl/planning/

[3] J.V.Gómez González, “Advanced applications of the fast marching square planning method”, Trabajo final de master, Dpto. de Ingeniería de Sistemas y Automática, Universidad Carlos III de Madrid, Leganés, Madrid, España, 2012. [En línea]. Disponible en: https://jvgomez.github.io/files/pubs/JVGG_Masters_Thesis.pdf

[4] A.Valero-Gomez, J.V.Gómez, S.Garrido y L.E.Moreno, “The Path to Efficiency: Fast Marching Method for Safer, More Efficient Mobile Robot Trajectories”, IEEE Robotics & Automation Magazine, vol. 20, n.º 4, pp. 111-120, Agosto 2013.

[5] S.Garrido, L.E.Moreno y J.V.Gomez. “Motion Planning Using Fast Marching Squared Method” en Motion and Operation Planning of Robotic Systems: Background and Practical Approaches. Researchgate, 2015. [En línea]. Disponible en: https://www.researchgate.net/publication/259715004_Motion_Planning_Using_Fast_Marching_Squared_Method

[6] V.Gónzalez Pérez, “Unmanned Aerial Vehicle Mission Planning Based on Fast Marching Square Planner and Differential Evolution”, Tesis doctoral, Dpto. De Ingeniería Eléctrica, Electrónica y Automática, Universidad Carlos III de Madrid, Leganés, Madrid, España, 2018. [En línea]. Disponible en: https://e-archivo.uc3m.es/bitstream/handle/10016/27741/tesis_veronica_gonzalez_perez_2018.pdf?sequence=1&isAllowed=y

[7] W.Bużantowicz. Matlab Script for 3D Visualization of Missile and Air Target Trajectories. ResearchGate, 2016. [En línea]. Disponible en: https://www.researchgate.net/publication/308324771_Matlab_Script_for_3D_Visualization_of_Missile_and_Air_Target_Trajectories

[8] W.Bużantowicz. Free Matlab package for 3d visualization of missile and air target trajectories. Wbint. [En línea]. Disponible en: http://www.wbint.pl/flypath3d/ (Acceso: noviembre 2019)

[9] Mathworks. “ROS Indigo and Gazebo”, Mathworks. [En línea]. Disponible en: https://es.mathworks.com/support/product/robotics/v3-installation-instructions.html

[10] Open Source Robotics Foundation. “What is a TurtleBot?”. Turtlebot. [En línea]. Disponible en: https://www.turtlebot.com/ (Acceso: diciembre 2019)

[11] Open Source Robotics Foundation. “Make a Mobile Robot”. Gazebosim. [En línea]. Disponible en: http://gazebosim.org/tutorials?tut=build_robot&cat=build_robot (Acceso: diciembre 2019)

[12] Open Source Robotics Foundation. “hector_quadrotor”. Wiki ROS. [En línea]. Disponible en: http://wiki.ros.org/hector_quadrotor (Acceso: enero 2020)

[13] E.Borghi. “AR.Drone-ROS”. GitHub. [En línea]. Disponible en: https://github.com/eborghi10/AR.Drone-ROS (Acceso: enero 2020)

[14] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, “RotorS—A Modular Gazebo MAV Simulator Framework,” in Studies in Computational Intelligence, vol. 625, Springer Verlag, 2016, pp. 595–625. [En línea]. Disponible en: https://github.com/ethz-asl/rotors_simulator

[15] Mathworks. “ROS Toolbox”. Mathworks. [En línea]. Disponible en: https://es.mathworks.com/help/ros/

[16] Coulter, R. Implementation of the Pure Pursuit Path Tracking Algorithm. Carnegie Mellon University, Pittsburgh, Pennsylvania, 1992. [En línea]. Disponible en: https://www.ri.cmu.edu/pub_files/pub3/coulter_r_craig_1992_1/coulter_r_craig_1992_1.pdf